Support and Rationale Document

for the

Software Communications Architecture Specification

MSRC-5000SRD

V1.2

December 21, 2000

Prepared for the

Joint Tactical Radio System (JTRS) Joint Program Office

Prepared by the

Modular Software-Programmable Radio Consortium

under Contract No. DAAB15-00-3-0001

Revision Summary

Ver. No.	Comment
1.0	Initial release in support of initial validation of the SCAS version 1.0
1.1	Release in support of SCAS version 1.1 with updates based on government comments to Version 1.0. This document was created to be in concert with Appendix A – Use Cases dated 9/15/00, Appendix B – Examples dated 6/30/00, Appendix C – Step 1 ADR dated 9/15/00 and Appendix D - Networking dated 6/30/00.
1.2	Add Appendix E, Rationale for the API Supplement v1.0.

Table of Contents

List of Illustrations

List of Tables

1 INTRODUCTION

The Software Communications Architecture (SCA) Support and Rationale Document (SRD) is a companion to the SCAS and provides background, tutorial, and support information for the development of Joint Tactical Radio System (JTRS) software configurable radios. For easier tracking of requirements and rationale, the SRD outline is similar to the SCAS in Sections 3 through 5. Together, the SCAS and SRD provide the framework, the rationale for that framework, and examples to illustrate the implementation of the architecture for differing domains/platforms and selected waveforms. Neither document specifies implementation details for any particular domain or waveform application.

1.1 SCOPE

This document provides:

1. Rationale explaining technical and methodology decisions made in the development of the SCAS.
2. Tutorial material and/or references on topics central to understanding the technologies involved in producing Modular Software Radio architecture and applications.
3. Background explanation of the requirements in the SCAS.

1.2 COMPLIANCE

All compliance rationale from this entire section is covered in the SCAS.

2 OVERVIEW

Section 2 of the SCAS contains a descriptive overview of the architecture as a framework for communications systems. Much of that descriptive material is derived from the Step 1 Architecture Definition Report (ADR) which appears in Appendix C to the SRD. Technical details and requirements of the architecture are contained in sections 3 through 5 of the SCAS and the corresponding rationales are contained in sections 3 through 5 of this document.

2.1 CORE FRAMEWORK CONCEPT

Core Framework is an architectural concept that defines a set of open application-layer interfaces and services to provide an abstraction of the underlying software and hardware layers for software application designers. All interfaces defined in section 3.1.3 of the SCAS were developed as part of the JTRS CF. Some of these are implemented by the CF Core Application Services (incorrectly, but forever to be called a "Core Framework"); others are implemented by non-core Applications (i.e. waveforms, etc.); one is implemented by device providers. Core Application Services are identified in Figure 2-1 in solid blue.

Organization of SCAS section 3.1.3 is intended to present the CF interfaces in logical groupings. It is not organized by requirement nor implementation, “CF vs. waveform application” and the groupings are independent of test method. The groupings selected are:

1. Base Application Interfaces (*Port*, *LifeCycle*, *TestableObject*, *PropertySet*, *ResourceFactory*, and *Resource*) are used by all waveform applications. SCAS 1.1 lists the required behavior for these interfaces. The associated operations are not optional but may be implemented as “NO-OP.”
2. Framework Control Interfaces (*Application*, *ApplicationFactory*, *DomainManager*, *Device*, and *DeviceManager*) provides control of the system. These control interfaces allow the deployment of waveform applications. This includes the setup and tear down of the waveform.
3. Framework Services Interfaces support both core and non-core applications (*File*, *FileSystem*, *FileManager*, *StringConsumer*, *Logger*, and *Timer*).

Each of the groups contain interfaces that must be implemented by waveform application and/or device developers. These include Base Application Interfaces, *Device* and *DeviceManager* (from Framework Control Interfaces), and *StringConsumer* (from Framework Services Interfaces). As shown in the CF Relationship figure 2-1.

Device and *DeviceManager* may be supplied by the CF provider with their Core Application Services as part of a system solution. Typically device developers will supply the Device and DeviceManager.

Waveform developers implement Base Application Interfaces. They do not implement the *Application* class, which is used for control, or any other Core Application Services.

Companies that purchase pieces of the system, System Implementers, must implement the *StringConsumer* interface as they integrate their system.

2.2 SCAS COMPLIANCE

For SCAS compliance certification, vendors must demonstrate SCAS compliance of their products.

Hardware vendors develop “hardware devices.” *Devices* (Logical) are created for “hardware devices.” The *Device* unit tests being developed for JTRS 2A can be used to validate device implementations against the SCAS.

Companies developing waveform applications must certify that their waveforms are SCAS compliant.

Companies providing Core Application Services must certify that their Core Framework Services (File Services, *Logger*, *DeviceManager*, if included, *DomainManager*, *Application*, *ApplicationFactory*) are SCAS compliant.

As a result of using SCAS compliant hardware devices, *Devices* and Applications, a company building a system can procure Core Application Services from company A, procure hardware devices and/or logical *Devices* from company B and procure Waveform Applications from company C while maintaining SCAS compliance.

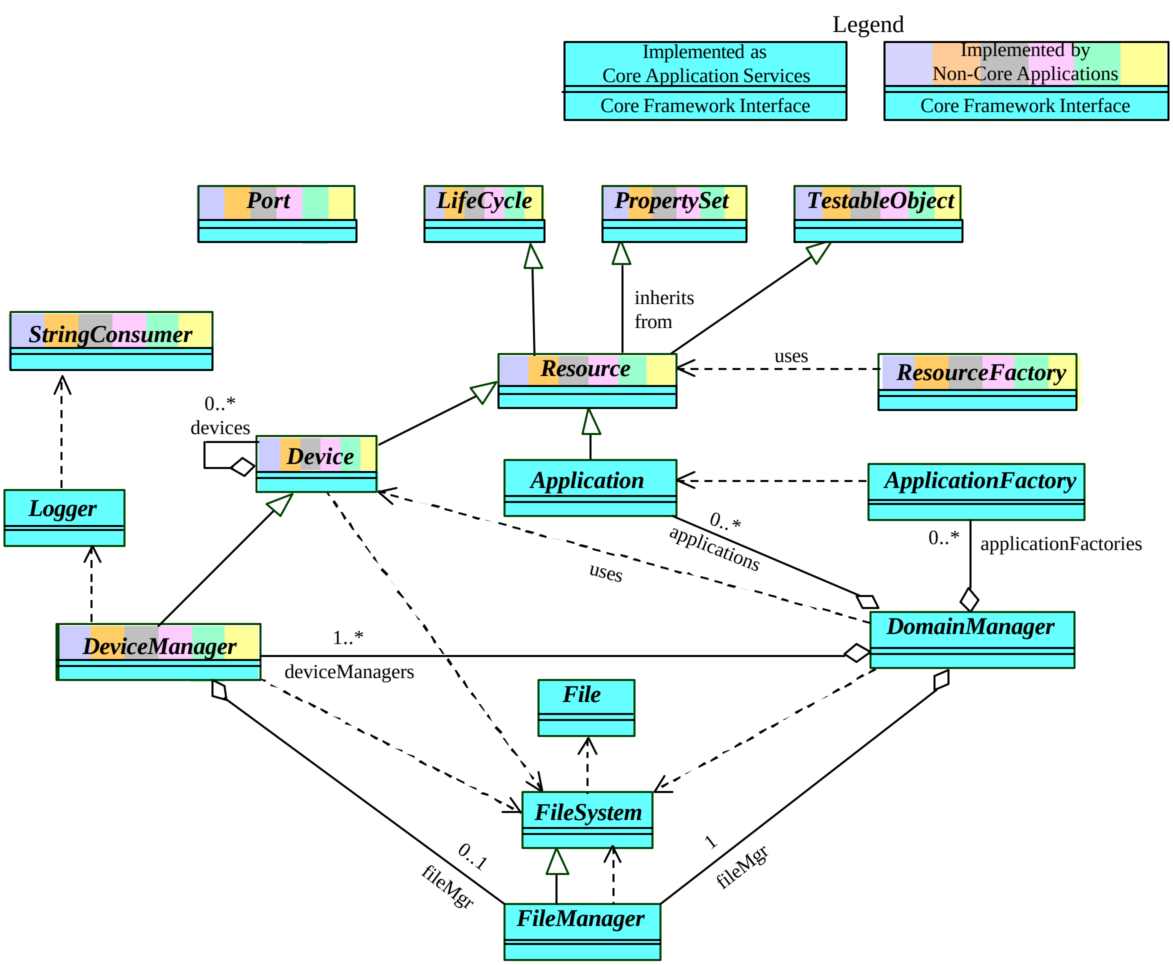

Figure 2-1. Core Framework Relationships

3 SOFTWARE ARCHITECTURE DEFINITION

3.1 OPERATING ENVIRONMENT

This section contains the rationale based on the requirements of the operating system, middleware, and the CF interfaces and operations that comprise the OE.

Paragraphs in section three of the SCAS many times address more than one requirement or concept at a time. Due to this, section three in this document reiterates some of the pertinent paragraphs from the SCAS (shown in bolded font). Rationale for each concept is then presented individually where appropriate.

There is also some rationale that does not fit exactly in a paragraph. In these cases, a next-level paragraph has been added.

3.1.1 Operating System

The processing environment and the functions performed in the architecture impose differing constraints on the architecture. An SCAS application environment profile (AEP) was defined to support portability of waveforms, scalability of the architecture, and commercial viability. POSIX services are used as a basis for this profile. See Figure 3-1. Notional Relationship of OE and Application to the SCAS AEP. The OS shall provide the services designated as mandatory by the AEP defined in Appendix B of the SCAS. The OS is not limited to providing the services designated as mandatory by the profile. The CORBA Object Request Broker (ORB) and the CF are not limited to using the services designated as mandatory by the profile.

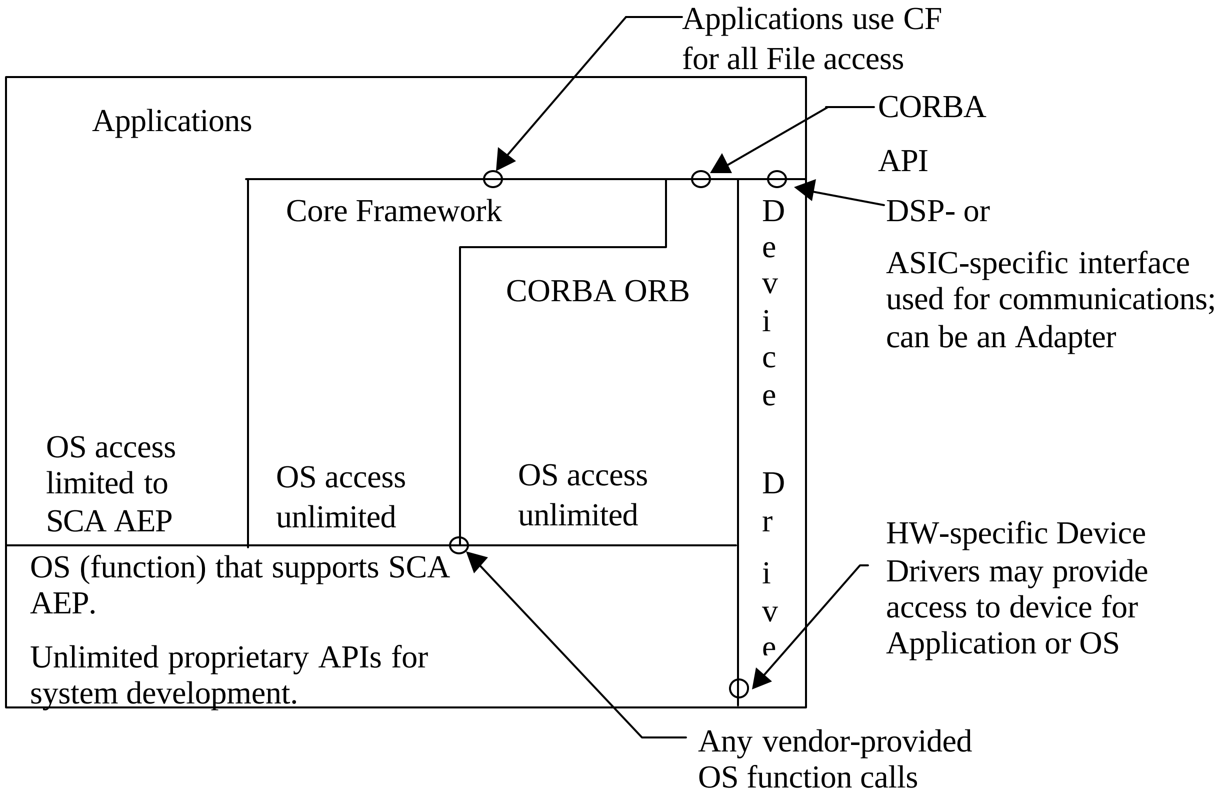

Figure 3-1. Notional Relationship of OE and Application to the SCAS AEP

3.1.1.1 POSIX

A key requirement for JTRS waveforms is portability. For waveforms to be portable, they will need to conform to a standard set of interfaces to the operating system services. The set of POSIX standards established by the IEEE defines such standard interfaces. POSIX is the only existing standard for real-time operating systems, (RTOS), interfaces, that is implemented wholly or in part by multiple operating systems and was therefore chosen as the basis for standardization of the SCAS OS interfaces.

3.1.1.2 POSIX 1003.13

The POSIX 1003.13 standard defines four Application Environment Profiles (AEP), PSE-51, PSE-52, PSE-53 and PSE-54. Each application profile defines the interfaces that must be implemented for the profile in terms of units of functionality in each of the POSIX.1, POSIX.1b, POSIX.1c and POSIX.5b standards. Each profile is designed to fit an application model. A brief description of the profiles follows.

3.1.1.2.1 Minimal Real-time System Profile (PSE-51)

This profile is a single process profile with no asynchronous or file I/O specified. Systems that implement this profile are typically embedded controllers.

3.1.1.2.2 Real-time Controller System Profile (PSE-52)

This profile is a single process profile that adds asynchronous and file I/O to PSE-51. Systems that implement this profile are typically embedded controllers.

3.1.1.2.3 Dedicated Real-time System Profile (PSE-53)

This profile adds multi-process capability to PSE-51.

3.1.1.2.4 Multi-Purpose Real-time System Profile (PSE-54)

This profile includes all the capabilities of the other three profiles and adds multi-user capabilities. The functionality includes all of POSIX.1, POSIX.1b and POSIX.1c and/or POSIX.5b

3.1.1.3 Choosing a Profile

One factor to consider when choosing a profile is that POSIX 1003.13 assumes no abstraction layer or middleware. The profiles are written such that for a given model applications will make direct calls to the OS to use certain services. With the existence of CORBA middleware and the CF abstraction layer in the architecture, some of the burden of achieving waveform portability can be shifted from the profile to those other elements.

An OS may be chosen for a given implementation of the SCAS and configured (read scaled) to accommodate a specific domain. Therefore, the SCAS must define a minimum set of POSIX functionality that the OS must provide in order to preserve waveform portability across domains while still providing the functionality required by waveforms. In addition the SCAS must in no way prohibit operating systems that implement additional features as they may provide needed or desired functionality for a given domain. Additional functionality provided, by the OS, must either be abstracted by the Core Framework or be transparent. In this way the architecture is allowed to scale from resource limited domains to full blown fixed installations. By limiting the applications to the SCAS defined profile, then maximum waveform portability will be ensured.

3.1.1.4 PSE-SCAS

The definition of a SCAS POSIX profile must allow a vendor of SCAS compliant radios to choose an operating system based on factors such as domain specific needs, features, cost, tool set and other factors. Since most operating systems, with guaranteed real time behavior, are of the single process type, the PSE-52 AEP defined in POSIX 1003.13 was chosen as a starting point. Although the PSE-52 profile is a single-process AEP, the SCAS modifies it and uses it in such a way as to allow multi-process, multi-user operating systems that may conform to PSE-54. In addition, PSE-52 includes file and file system capability which the SCAS needs. This modified AEP is called PSE-SCAS. Since this profile does not require a particular memory model it allows for different memory protection schemes such as the process model as typified by UNIX an its derivatives and protection domains which have been implemented in experimental operating systems such as Pebble[1] and Opal[2].

VxWorks is the most popular real-time operating system and a good example of a single process operating system. The profile therefore was tailored to allow VxWorks which may be suitable for certain domains. Wind River's VxWorks operating system was selected as one of two Real Time Operating Systems (RTOS) implementations because of the following reasons:

1. It is an embedded real time OS
2. It is POSIX compliant, an indirect JTA requirement via DII COE level 2
3. It is supported by MOST COTS control processor boards
4. It provides a rich tool set for software development
5. It provides a rich interface support set: TCP/IP, UDP, BUS, Ethernet, SLIP, PPP, SNMP, ftp, telnet, sockets, CORBA, RPC
6. It is currently being used in Raytheon legacy digital radio products: WNR, ACN, ACR, Raytheon's DMR, JTT

Lynx's LynxOS operating system was also selected as one of two Real Time Operating Systems (RTOS) implementations because of the following reasons:

1. It is an embedded real time OS
2. It is POSIX compliant, an indirect JTA requirement via DII COE level 2
3. It is supported by MOST COTS control processor boards
4. It provides a rich interface support set: TCP/IP, UDP, BUS, Ethernet, SLIP, PPP, SNMP, ftp, telnet, sockets, CORBA, RPC
5. It is currently being used by Rockwell's legacy digital radio product

One of the goals of the SCAS is to achieve maximum portability for applications. If an OS provides interfaces and functionality not specified in the AEP and an application makes use of the functionality by making direct calls to the OS, then that application would have to be modified to port it to a platform with an OS that did not support the additional functionality. To

---

[1] The Pebble Component-Based Operating System

Eran Grabber, Christopher Small, John Bruno, Jose' Brustoloni and Avi Silberschatz

http://www.bell-labs.com/project/pebble/

[2] Sharing and Protection in a Single-Address-Space Operating System

Jeffrey S. Chase, Henry M. Levy, Michael J. Feely, and Edward D. Lazowska

ACM Transactions on Computer Systems, 12(4), November 1994 achieve the goal of maximum portability, applications must therefor be limited to the PSE-SCAS AEP.

3.1.1.4.1 Application conformance to PSE-SCAS

One of the goals, of the SCAS, is to achieve maximum portability for applications. If an OS provides interfaces and functionality not specified in the AEP and an application makes use of the functionality by making direct calls to the OS, then that application would have to be modified to port it to a platform with an OS that did not support the additional functionality. To achieve the goal of maximum portability, applications must therefore be limited to the PSE-SCAS AEP.

3.1.1.4.2 OS conformance to PSE-SCAS

If applications must adhere to the PSE-SCAS AEP, then it follows that the OS must provide the interfaces and functionality that the AEP specifies. The difference is that the AEP is a maximum for an application and a minimum for an OS. That is, the OS in an SCAS compliant platform must at a minimum provide the interfaces and functionality that the AEP specifies. The OS can provide additional functionality either through vendor specific interfaces or through additional POSIX interfaces including full PSE-54 conformance. The term OS, in this context, includes third party vendor provided interfaces and functionality. For instance, software from a third party vendor can provide the pthread interfaces specified in the AEP to abstract OS vendor specific interfaces.

3.1.1.4.3 CF conformance to PSE-SCAS

The CF is not restricted to the PSE-SCAS. Although it may be desirable for the CF to be portable across platforms and operating systems it is not necessary. In fact, to achieve maximum portability of applications across operating systems, it is necessary to give up some portability in the CF to achieve the more important goal of application portability. The CF, in this context, means functionality with CF interfaces that is provided as part of the OE. In other words, applications that provides services with CF defined interfaces are still considered applications and not part of the OE and therefore must adhere to the AEP. The Domain Manager, File Manager, Device Managers and Devices are considered part of the OE and are not restricted to the AEP.

3.1.1.4.4 Deviations from PSE-52

3.1.1.4.4.1 POSIX.1

3.1.1.4.4.1.1 POSIX_SINGLE_PROCESS Functions

3.1.1.4.4.1.1.1 uname

The uname function retrieves the system name, node name, OS release and version and machine (hardware identifier). This function was omitted from PSE-SCAS for the following reasons:

1. It is not consistently implemented across operating systems
2. Identity is established by the CF::Device element, which can use OS specific calls.

3.1.1.4.4.1.1.2 sysconf

The sysconf function provides a method for an application to determine the current value of a configurable system limit or option. This function was omitted from the profile for the following reasons:

1. It is not consistently implemented across operating systems
2. The SCAS explicitly specifies which features must be supported and therefore dynamic reconfiguration based on the values of sysconf flags is not an option.
3. Many of the sysconf limit values pertain to the file system and the SCAS standardizes the file system by requiring that all file access go through the CF.
4. Other limit values pertain to some resource capacity that may be addressed by the CF::Device.

3.1.1.4.4.1.2 POSIX_MULTI_PROCESS Functions

3.1.1.4.4.1.2.1 setlocale

This function is not a strictly multi-process function and was deemed to be of some usefulness for foreign radios.

3.1.1.4.4.1.3 POSIX_SIGNALS Functions

3.1.1.4.4.1.3.1 alarm

The alarm function is process oriented. It sends signals to a process and not a thread. Since some RTOSs do not implement processes they do/will not implement alarm. The alarm time granularity is seconds which is of limited utility in real time environment such as a radio. In addition, the functionality provided by alarm is also provided by POSIX timers who have a much higher time resolution.

3.1.1.4.4.1.3.2 sigsetjmp, siglongjmp

These functions were omitted because they are not implemented consistently across operating systems. If the purpose is to jump from a signal handler to a cleanup code, setjmp and longjmp will suffice.

3.1.1.4.4.1.4 POSIX_FILE_SYSTEM Functions

The SCAS states that although file system functions are included in the AEP, applications are prohibited from using them directly. The reasoning behind this was that the file access functions are implemented widely in operating systems and therefore there would be no penalty in including them in the profile and requiring an operating system to provide them. The CF could use these functions and be more portable.

The CF is designed to provide location transparency for file access with CORBA serving as the distribution mechanism just as remote procedure calls (RPC) do for the Sun Networked File System (NFS). To maintain transparency the applications must therefore be restricted to the CF.

3.1.1.4.4.1.5 POSIX_FD_MGMT Functions

3.1.1.4.4.1.5.1 dup

This function is normally used for inter-process communication, (IPC), (i.e. pipes). Single address operating systems do not implement processes and the responsibility for IPC lies with the CF. This function was therefore omitted from the profile.

3.1.1.4.4.1.5.2 dup2

This function is normally used for redirection of stdin and stdout. Applications are restricted to using the CF for file I/O for the reasons stated in section 3.1.1.4.4.1.4.

3.1.1.4.4.1.5.3 fcntl

Applications are restricted to using the CF for file I/O for the reasons stated in section 3.1.1.4.4.1.4.

3.1.1.4.4.1.6 POSIX_DEVICE_IO Functions

Adapters may need to communicate with devices directly so applications may use these functions. Applications may not use these functions for file I/O for the reasons stated in section 3.1.1.4.4.1.4.

3.1.1.4.4.1.7 POSIX_C_LANG_SUPPORT Date and Time Functions

3.1.1.4.4.1.7.1 tzset

The setting of the time zone is more properly a global function within a radio system or a LAN. Applications, as defined in JTRS should not be setting the time zone. The time should be set on a workstation on a LAN or through the CF (i.e. time service).

3.1.1.4.4.2 POSIX.1b

3.1.1.4.4.2.1 _POSIX_MAPPED_FILES

Mapped files can be used to implement IPC or to persistently store data using memory access semantics in a language. The IPC mechanism in the SCAS is CORBA, which allows for transparent distribution of applications and objects. Persistent storage should be accomplished through the CF file system.

3.1.1.4.4.2.2 _POSIX_SHARED_MEMORY_OBJECTS

Shared memory is used to communicate between processes in a multi-process operating system without incurring the overhead of pipes. Processes may be distributed across processors. Using this capability directly prohibits location transparency of the processes. The IPC mechanism in the SCAS is CORBA, which allows for transparent distribution of applications and objects. If efficiency is a concern, a pluggable transport for CORBA that uses shared memory may be used as CORBA implementations are not restricted to the profile.

3.1.1.4.4.2.3 _POSIX_SYNCHRONIZED_IO

3.1.1.4.4.2.3.1 fdatasync

Applications are restricted to using the CF for file I/O for the reasons stated in section 3.1.1.4.4.1.4.

3.1.1.4.4.2.4 _POSIX_FSYNC

3.1.1.4.4.2.4.1 fsync

Applications are restricted to using the CF for file I/O for the reasons stated in section 3.1.1.4.4.1.4.

3.1.1.4.4.3 POSIX.1c

3.1.1.4.4.3.1 POSIX_FILE_LOCKING functions

Applications must use the CF for file operations. The CF will have the responsibility for controlling mutually exclusive access to files.

Applications are limited to using the OS services that are designated as mandatory for the profile. Applications will perform file access through the CF. (Application requirements are covered in section 3.2.)

3.1.2 Middleware & Services

3.1.2.1 CORBA

1.

3.1.2.2 CORBA Extensions

At the time of writing of this document, the CORBA extensions have not been fully investigated yet to determine if they should be included in the SCA as mandatory.

3.1.2.2.1 Naming Service

At the time of the writing of this document, the Naming Service is a required extension for the CF.

ORBExpress has delivered a beta version of a target naming service for VxWorks on the x86 and PPC platforms. It is being tested at the time of the writing of this document.

3.1.2.2.2 Quality of Service Control

Both the Real-Time and Messaging CORBA extensions provide features that are commonly used or needed in embedded communications systems.

3.1.2.2.2.1 Real-time

ORBExpress has delivered a beta version of a Real-time ORB for VxWorks on the x86 and PPC platforms. At the time of the writing of this document, none of the Consortium members have attempted to use this beta version of a Real-time ORB.

The Real-Time CORBA extension as specified by the OMG Document orbos/98-12-05, December 21, 1998 provides scheduling & priority capabilities along with other real-time features (e.g., mutex, communication protocols, multiple transport connections, thread pooling). The Real-Time CORBA specification is based upon POA plus the CORBA Message Specification.

3.1.2.2.2.2 Messaging

The CORBA Messaging as specified by the OMG Document orbos/98-05-05, May 18, 1998 provides:

Asynchronous method invocation model in CORBA

Quality of service (QoS) features associated with messaging systems such as the typical Message Oriented Middleware (MOM) products. These QoS features include: Delivery Quality, QueueManagement and Message Priority.

Interoperable store-and-forward capability in IIOP. This defines a Routing protocol along with several extensions to GIOP:

- Invocations on targets which are not currently active or cannot be activated. These requests are delivered to the targets when the appropriate store and forward mechanisms allow, subject to the lifetime of the request specified by the QoS.

- Replies to clients which are not currently active or cannot be activated. Replies are delivered to clients when the appropriate store and forward mechanisms allow, subject to the lifetime of the request specified by the QoS.

At the time of the writing of this document, none of the Consortium members have attempted to use CORBA Messaging extensions. Therefore, this extension may be considered for removal from the SCA at a later date. However, this decision should be held until the architecture has been proven.

3.1.2.3 CORBA Implementations

Objective Interface System Inc.'s ORBExpress RT CORBA was selected as one of two RTOS ORB implementations s because of the following reasons:

1. It was the only COTS OMG version 2.2 compliant CORBA ORB utilizing the Portable Object Adapter (POA) on the market as of 02/01/2000
2. It provides a language independent IDL compiler
3. It is COTS
4. It is supported by Wind River's VxWorks and Lynx's LynxOS RTOS
5. It is available for x86 and Power PC platforms.
6. It supports CORBA Naming Services
7. It was selected by MSRC consortium members for their prototype radios.

HighComm's VisiBroker CORBA was selected as one of two Real Time ORB implementations because of the following reasons:

1. It provides a language independent IDL compiler
2. It is COTS
3. It is supported by Wind River's VxWorks RTOS
4. It is available for x86 and Power PC platforms.
5. It supports CORBA Naming Services

It is currently being used in Raytheon legacy digital radio products: WNR, ACN, ACR, Raytheon's DMR

3.1.3 Core Framework

3.1.3.1 Base Application Interfaces

3.1.3.1.1 *Port*

The *CF Port* interface is used to support the Software Profile's Software Assembly Descriptor in setting up or tearing down connections between CORBA components. This interface both uses and provides ports when an application is created or terminated. *Port* has evolved from the MessageConsumer and MessageProducer interfaces of SCAS 0.3. These interfaces had evolved from the JTRS Step 1 MessageRegistration and Message interfaces. Advantages of the *Port* interface over the previous interfaces are:

1. The interface supports the connection concepts in the CORBA Components Specification where any provides port type can be connected to another uses port, as long the uses port understands the port it is connecting to.
2. The uses port can be behave either as a push producer or a pull consumer. Likewise, the provides port type can be behave either as a push consumer or pull producer.
3. The implementation of a port allows for fan-in or fan-out implementations.
4. Specific interfaces can be developed for a port, thus eliminating no-op operations when an all encompassing interface is used (e.g., Message, MessageConsumer).
5. A simple and specific interface.

ApplicationFactory and *Application* implementations use the *Port* interface. Application developers implement the *Port* interface for its uses port. An application uses port must be a *CF Port* Type. The uses and provides ports are paired up in the Software Profile's Software Assembly Descriptor which are known as a connection interface elements which are contained in the connections element. In the Software Profile's Software Component Descriptor, a port can only support one type of interface since the *connectPort* operation doesn't indicate the type of interface being used. The *connectPort* operation takes in a CORBA object reference that must be narrowed to the interface the uses port supports. This generic operation allows a *CF ApplicationFactory* and *CF Application* implementations to setup or tear down connections between any Application CORBA software components. The name parameter for the CF Port operations is a unique connection identifier created by the *CF ApplicationFactory* at the time a connection is created between uses and provides port. This supports generic fan-in and fan-out implementations without the uses or provides ports actually knowing the specific ports connected to it.

3.1.3.1.2 LifeCycle

3.1.3.1.2.1

The *Lifecycle* interface is a generic way of initializing and releasing a resource, application, or device within the domain. These interface operations came from the JTRS Step 1 StateManagement interface which were pulled out into a separate interface, thus allowing for reuse for software components that need this behavior. This concept is an agreed upon concept from the SDR forum.

ApplicationFactory implementations use this interface to initialize an application after all ports have been connected together.

HCI uses this interface to release an application or a device. *Application* implementations use this operation to release application's components.

3.1.3.1.3 *TestableObject*

3.1.3.1.3.1

The *TestableObject* interface is a generic way of testing a resource, application, or device within a domain. Test descriptions, along with their results, are described in the Software Profile's Property File.

These interface operations came from the JTRS Step 1 StateManagement interface which were pulled out into a separate interface, thus allowing for reuse for software components that need this behavior. This concept is an agreed upon concept from the SDR forum.

ApplicationFactory implementations use this interface to test an application. Currently this capability is not a requirement in the SCAS.

TestableObject is also used by an HCI to test an application or device within the domain. With this interface, it would be possible to have a generic HCI that could work for all Applications.

TestableObject can be used for testing remotely over the air without operator intervention.

3.1.3.1.4 *PropertySet*

3.1.3.1.4.1

The *PropertySet* interface is used to support the Software Profile's Property File. Properties, (Id(name)/value pairs), are concepts from the CORBA Components and TINA specifications. These interface operations came from the JTRS Step 1 StateManagement interface which were pulled out into a separate interface, thus allowing for reuse for software components that need this behavior. This concept is an agreed upon concept from the SDR forum.

ApplicationFactory implementations use this interface to initially configure an application.

PropertySet can be used by an HCI to configure an application or device within the domain. With this interface, it is possible to have a generic HCI that could work for all Applications.

PropertySet can be used for over the air remote configuration without operator intervention. Or, query for information remotely over the air from any application within the domain.

3.1.3.1.5 *Resource*

The *Resource* interface is based upon a minimal set of interfaces needed to control a component. That is, a resource, device, or an application after it has been created and services are needed by the *ApplicationFactory* to initialize and configure it. *Resource* is basically the same as stated in JTRS Step 1 but with fewer operations (elimination of MessageRegistration, Message interfaces, state operations and attributes).

Resource is used by an HCI to configure an application or device within the domain. With this interface, it would be possible to have a generic HCI that could work for all Applications.

The *getPort* operation from *Resource* supports the Software Profile's Software Assembly Descriptor. *getPort* is used by the *ApplicationFactory* to retrieve uses and provides ports, and supporting interfaces, which are interfaces that are inherited. figure 3-2, depicts a resource which has three provides (IN_1, IN_2, Resource) ports and one uses (out) port.

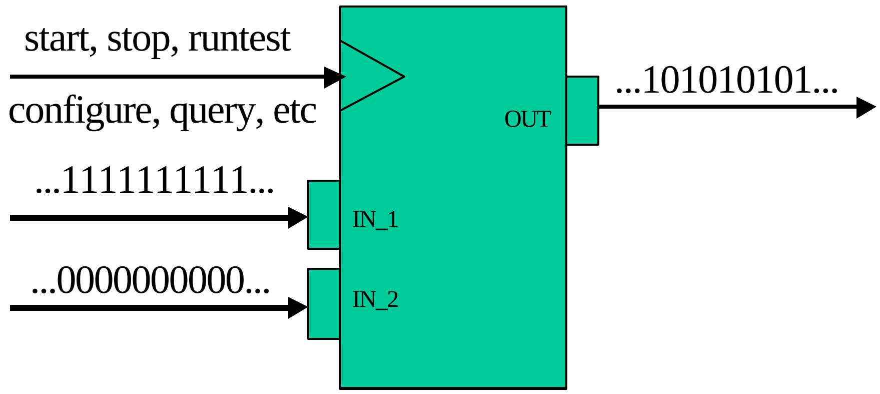

Figure 3-2. *Multiplexer* Resource

3.1.3.1.6 *ResourceFactory*

The *ResourceFactory* interface class provides a means to request the creation or destruction of a specific type of *Resource* in the domain. *ResourceFactory* was designed after the Factory Design Pattern.

The Software Profile determines the type of *Resource* that is created by the *ResourceFactory*. Properties passed to the *ResourceFactory create* operation indicate the type of *Resource* to be created.

ResourceFactory provides a design and implementation trade off on whether Resource instances need to be in their own OS Process Space or in the same OS Process Space. A Resource can be executed on the same processor multiple times or can be created multiple times using the *ResourceFactory*.

3.1.3.1.6.1 Optional use of *ResourceFactory*

ResourceFactory is desirable when used in a multi-channel radio. ResourceFactory allows the DomainManager to instantiate multiple copies of the same application waveform without reloading each time.

3.1.3.2 Framework Control Interfaces

In the problem space, a Domain consists of SW Applications (installed services), Nodes and their devices, file manager, and SW Application Instances as depicted in figure 3-3.

A Node is an abstraction of the computing node, that allows the control and monitoring of the node resources (CPU, processes, memory and file systems), Implements the OE, executes a

DeviceManager and installed OE & CF services (e.g., CORBA Naming Service, Logger, FileSystem, etc.).

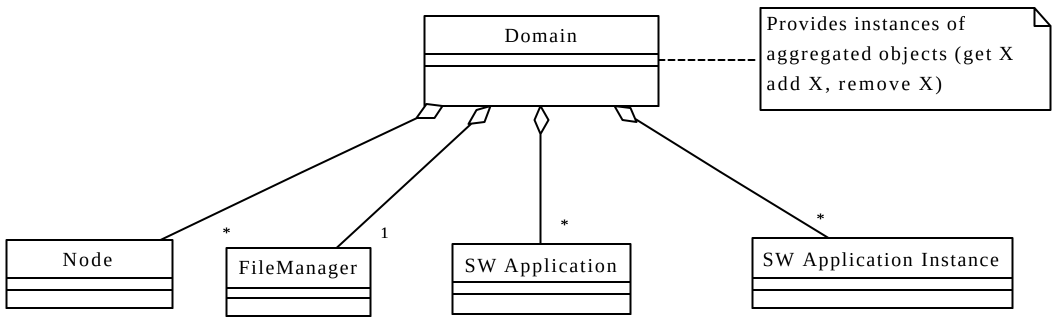

Figure 3-3. Domain In Problem Space

The services provided by a Domain are:

1. Installing and uninstalling application software onto the domain's file manager.
2. Retrieving Nodes, SW Applications, and SW Application Instances.
3. Creating, Terminating, and Controlling SW Application Instances.
4. Registering and unregistering Nodes along with their devices.

These Domain services are mapped to three CF interfaces:

1. *DomainManager* provides the services for retrieving, installing/registering, and uninstalling/unregistering Domain elements. An issue has been posted about breaking this interface into three interfaces.
2. *Application* provides the services for terminating and controlling the SW Application Instance.
3. *ApplicationFactory* provides the services for the creation of an Application. For each SW Application that is installed in the Domain, an ApplicationFactory object is created that is used for deploying the application within the Domain.

3.1.3.2.1 *Application*

3.1.3.2.1.1

From an end-user perspective, software applications (services, waveforms, etc.) are the primary functional element of a JTRS radio. Since users need to control the lifespan of applications, configure them, and control their behavior, the *Application* interface provides the necessary operations for managing application lifecycle, state, and behavior.

An *Application* has characteristics very similar to a *Resource*. For example, an application has properties and communicates through ports. Therefore, the *Application* interface is derived from the *Resource* interface. In general, the implementation of an *Application* will be comprised of a set of *Resource* implementations and their interconnections. The Application interface will then be a subset of the interfaces defined for its resources. In addition, the *Application* interface provides the filename containing the XML profile information that describes its internal structure and the characteristics of its properties and ports.

The Application interface provides a standard interface for all applications within the domain, and one common implementation (although each CF implementation may be different) within the domain for all applications. Each application developer doesn't have to develop code to tear down their application and to behave according to the application software profile (SAD).The *Application* ports are intended to be used for user interfaces or for connecting applications. Having a generic application abstraction supports general-purpose user interface components for controlling a JTRS radio.

3.1.3.2.2 *ApplicationFactory*

3.1.3.2.2.1

Although the *Application* interface provides the operations for controlling most of an application's lifecycle, it cannot be responsible for the actual creation of an application (since it doesn't exist yet). For a user to start an application, they must be aware of the types of applications available in the radio and then be able to command the creation of a new instance of a selected application type.

The *ApplicationFactory* interface class provides an interface to request the creation of a specific type of application in the domain. This abstraction allows a user interface to manipulate a group of application types as objects. The user (through the user interface) can create an application instance, provide it with pre-selected devices, and specify it configuration properties. The interface also provides a user-friendly name for the application type and provides access to the software profile used for creating applications of that type. Figure 3-4, Example ApplicationFactory, illustrates how the *ApplicationFactory* interface is used to bring a new Application instance into existence.

The *ApplicationFactory* interface class is designed using the Factory Design Pattern. There will be one *ApplicationFactory* implementation for each type of application.

The ApplicationFactory interface provides a standard interface for creating applications within the domain, and one common implementation (although each CF implementation may be different) within the domain for creating applications. The ApplicationFactory creates a CORBA object implementing the Application interface for each application created. Each application developer doesn't have to develop code to parse their own software profiles and retrieve domain profile (devices, etc.) to create their application within a domain. An ApplicationFactory CORBA object (implementing the ApplicationFactory interface) is created for each different Software Assembly Descriptor (SAD) XML installed in the domain. The ApplicationFactory is the technique used for creating an application after a domain has started up or for domain automatic applications start up. For domain automatic applications start up, the application software profiles (SADs) have to already exist within the domain. An application can be one or more resources, where the same resource components may be reference by multiple different application software profiles (SADs), thus allowing aggregate applications to be formed up. The

ApplicationFactory also provides the mechanism for future application creation control (e.g., disable, enable).

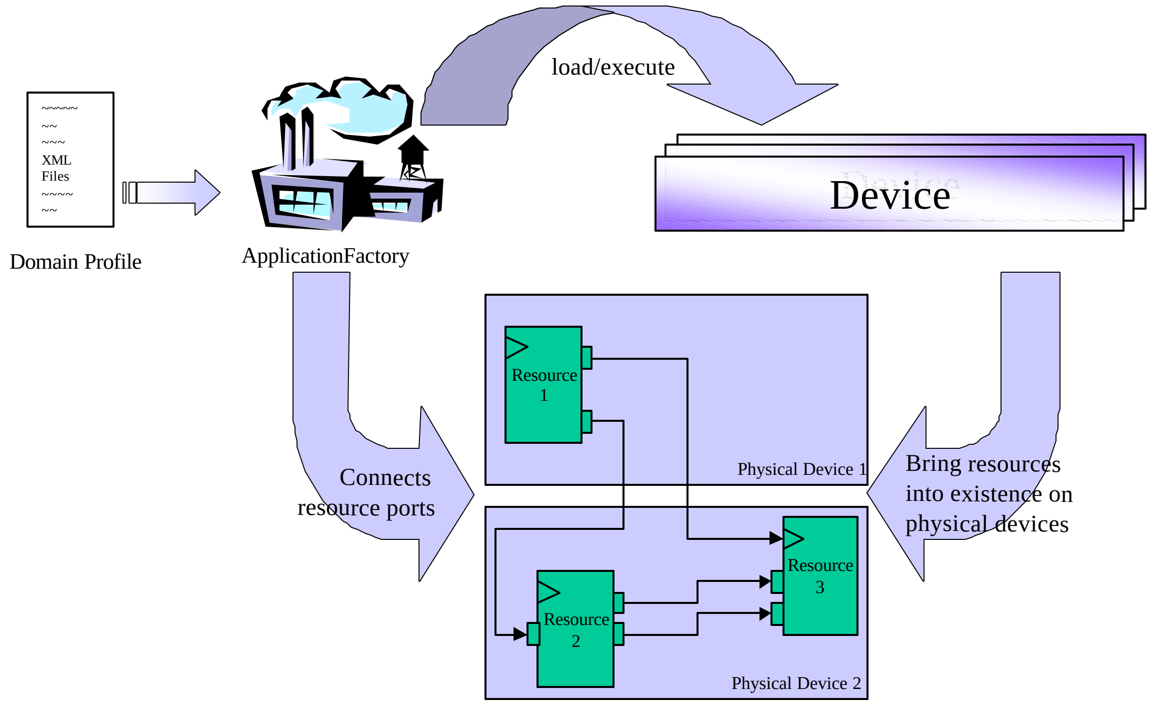

Figure 3-4. Example ApplicationFactory

3.1.3.2.3 Domain Manager

The Domain Manager, as described in the SCAS, has the responsibility to determine the proper allocation, de-allocation, and operation of the software components and Devices that make up the communications system termed the JTRS. The Domain Profile receives information from various elements in the architecture, including all the Device Managers (Device Profiles), and the Install Service (Application Profiles).

By choosing a format that is an open standard for the import/export of the Application, and Device Profile information; the specification will assure the interoperability and ease of delivery of components to the architecture. Waveform vendors, hardware component, and software component vendors will all use a standard means to delivery interoperable information to the JTRS developers.

The eXtensible Markup Language (XML) has become a very important element in many Object-Oriented Framework designs recently, including being a critical element in the recently ratified CORBA Components Model of the Object Management Group (OMG). XML has been continuing to grow wide spread support in both documentation and software development tools support over recent months. XML format has excellent flexibility that makes it ideal for a standard that will require multi-vendor support like the SCAS Architecture. The Document Type Definition (DTD) defines the interpretation of a specific set of tags that describe the Application, Resource and Device characteristics needed for Domain Management.

Other XML benefits include: Readability and clarity; support of object-oriented technology and Component-based architecture (structured storage of information); Extensibility to support additional migration of capabilities; Flexible Metadata, Name-valued pair capabilities; Supports Standard Mechanism for documenting important features of the Architectural Components; Supports CM Tasks necessary for definition of SDR-specific standards.

3.1.3.2.4 *Device*

3.1.3.2.4.1

Device interfaces provide an abstraction of the hardware elements that software components require. Software components (*Resources*) are associated with logical devices. This association can take two forms. In the first form, a *Resource* can be hosted on devices (a CPU, for example) or second use a device (a GPS receiver, modem, etc.). *Device* supports configuration and control and allows devices to be associated with *Resources*. The load and execute operations, of the *Device* interface, support the “hosted on” association, while the port interface, derived from the *Resource* interface, provides the “uses” association.

Since the *Device* interface merely provides an abstraction of hardware, it may be the case that many *Devices* can exist per physical hardware element. A device such as a programmable modem could present itself as both a CDMA and a TDMA modem simultaneously.

This abstraction also allows physical devices to be emulated. For example, a Device such as a printer could be emulated on a general-purpose processor. Instead of providing a printed page of the information sent to it, the printer Device could present the page graphically on a display.

3.1.3.2.5 *DeviceManager*

3.1.3.2.5.1

The *DeviceManager* interface serves to provide a container for *Device* objects that it knows about. Typically, a *DeviceManager* can be found on each CORBA-capable processor in the radio.

Since the *DeviceManager* inherits from the *Device* interface, it is intended that the *DeviceManager* represent the CORBA-capable processor on which it is running.

3.1.3.2.5.2 DeviceManager Interface

The *DeviceManager* interface provides the ability to install new *Device* objects into the radio dynamically and provides the means by which its *Logger* and *FileManager* can be found.

During the DeviceManager review (02/0700 through 02/12/00), it was decided to break out software operations into a new interface known as the Device. The benefits for the Device interface are:

better OO design, more distributed – Software development and maintenance are improved given this separation of functionality into separate classes. Development is improved through isolation of cohesive functions and use of inheritance. Well-defined interfaces provides encapsulation and easier future extensibility.

no central point of failure – Previous choice of including Device functionality within the Device Manager would be more susceptible to subsystem failure due to single device problems.

narrow applications scope– Application developers need only see the Device interface and not the full Device Manager.

applications don't have access to the DeviceManager – This improves the system security and simplifies the application.

The Device interface does however add more CORBA overhead.

3.1.3.3 Framework Services Interfaces

3.1.3.3.1 File

Applications must use the CF file interfaces so that the location of the files is transparent to the application. The applications can access all files using the CORBA File interface whether the file is local or remote. This provides consistency and portability of applications.

See section 3.1.1.4.4.1.4 POSIX_FILE_SYSTEM Functions for further description of why POSIX file commands were replaced.

3.1.3.3.2 ***FileSystem***

3.1.3.3.2.1

Network File Systems, (NFS), may not be resident. The CF supports a networked file system paradigm (CORBA taking the place of remote procedure call, RPC.) To transparently access files across platforms, the CF File, FileSystem, and FileManager interfaces must be used.

See section 3.1.1.4.4.1.4 POSIX_FILE_SYSTEM Functions for further description of why POSIX file commands were replaced.

3.1.3.3.3 ***FileManager***

3.1.3.3.3.1

Network File Systems, (NFS), may not be resident. The CF supports a networked file system paradigm (CORBA taking the place of remote procedure call, RPC.) To transparently access files across platforms, the CF File, FileSystem, and FileManager interfaces must be used.

See section 3.1.1.4.4.1.4 POSIX_FILE_SYSTEM Functions for further description of why POSIX file commands were replaced.

3.1.3.3.4 *StringConsumer*

3.1.3.3.4.1

No further explanation required.

3.1.3.3.5 *Logger*

3.1.3.3.5.1

No further explanation required.

3.1.3.4 Domain Profile

The elements of the Domain Profile have been based upon the OMG's CORBA Components specification (orbos/99-07-01). The CORBA Components specification was identified as a good starting point to base the SCA Domain Profile on because it contained much of the functionality required by the SCA and it was already established as an OMG draft. Many of the elements of the Domain Profile described below were derived from the corresponding Component Model and then modified to fit the needs of the SCA. This approach was taken so that we could build upon the work already completed by the OMG and not duplicate it.

3.1.3.4.1 Software Package Descriptor

This definition is derived from the CORBA Components Software Package Descriptor. The software package is used to describe any software component (non-CORBA, CORBA, etc.). The general information about a software package consists of name, author, properties, software component and implementation elements. The Implementation element describes the code dependency on hardware (hardware class and hardware capacities needed) and software, object code file reference, and the type of code element to be loaded. A Software package may have many different implementations (on different hardware) using the same properties and Software Component Descriptor. The Software Component Descriptor element defines the “provides” and “uses” interfaces (ports) for a CORBA capable component.

3.1.3.4.2 Software Component Descriptor

This definition is derived from the CORBA Component Descriptor file definition (interfaces, provides port, uses port, supporting interface). The SCAS components (*Resource*, *ResourceFactory*, *Device*), like the CORBA Components Specification, have CORBA capable components with well-defined interfaces and both have components that use well-defined interfaces. Like the CORBA Components Specification, component interfaces can be described in two ways thus providing design and implementation flexibility:

1. Supporting interfaces which are IDL interfaces inherited by the component IDL interface.
2. Provides interfaces, which are known as facets that are distinct CORBA interfaces that a component implementation supports besides a component’s interface.

3.1.3.4.3 Software Assembly Descriptor

This definition is derived from the CORBA Component Assembly Descriptor (CAD) file definition (partitioning, connections, component files). The Software Assembly Descriptor, (SAD), like the CORBA Components Specification CAD is used to:

1. Describe what component instances makeup an assembly.
2. Describe which components are located together.
3. Describe how to find components (naming service, stringified IOR, etc.).
4. Describe how to connect components together.

Other features added to the SAD are:

1. The optional identification of a component as the main assembly controller.
2. The optional visible ports for a software assembly

3.1.3.4.4 Property File

This definition is derived from the CORBA Components Properties element. Only the simple element definition was used from the CORBA Components Properties element. The elements added to the properties simple element definition are:

1. Mode – identifies the access control for a property. This element is only effective when the kind is “configure”.
2. ID – identifies an unique identifier string for a property. The name element is optional. When an ID is a meaningful value, (like a name) then the simple name attribute is not necessary.
3. Enumeration – identifies a list of enumerations with assigned values.
4. Range – identifies min and max values for a property.
5. Units – identifies the unit of measure for a property.
6. Kind – identifies the kind of property. A property can be multiple kinds or used for multiple purposes. For example, to be able to query an executable property type, the kinds would be configure (and mode = readonly) and execparam.

A simplesequence element is also defined which is based upon the simple element definition. The difference between a simple and simplesequence elements is the simple element only has one value where a simplesequence element can have a list of values. Each simplesequence value has the same units, range, and type. A Property File contains information about the properties applicable to a software package or a software component, such as frequency and power level. A Device Property File contains information about the properties of a device, such as processor types and device capacities. Properties defined in the XML can be set and retrieve using the *CF PropertySet* interface, run a test using the *CF Testable* interface, and as execute parameters using the *CF Device* interface

3.1.3.4.5 Device Package Descriptor File

This definition is derived from the CORBA Components Software Package Descriptor but without the Implementation and CORBA Component Descriptor elements. A Device Package Descriptor File identifies a class of a device along with its properties. Device identification information, including the device name, device class, model number, and serial number are contained in a Device Package Descriptor File.

3.1.3.4.6 Node Configuration Descriptor File

This definition is derived from the Software Assembly Descriptor (SAD) file definition (partitioning, connections, component files). The Node Configuration Descriptor (NCD) is used to describe the configuration for a device. The NCD can be used to alter the configuration of a device. The NCD like the SAD is used to describe what component instances (devices, CF services, etc.) get created for a device configuration. Specific NCD features are:

1. Describe how to obtain the *CF DomainManager* component object reference (naming service, stringified IOR, etc.).
2. Describe file system mount point names (optional).
3. Describe whether a CF Logger should be created or how to obtain a CF Logger object reference.

3.1.3.4.7 Profile Descriptor

This definition is used to identify a file name for a SAD, SPD, or NCD. The file name is a full pathname that includes a mounted file system name. This profile XML element can be returned by a CF interface's (*CF Device, DeviceManager, Application, and ApplicationFactory*) profile attribute. The profile attribute at one time use to be a CF File, but this was changed to a string type since a device may not have a file system, therefore don't force the device to create a CF File object for a profile. The profile attribute string definition was to allow to either contain a profile file path name or the full XML definition for a SPD, SAD, and NCD. This XML definition was chosen for the profile file path name in order to give out XML information in both cases for a profile attribute.

3.2 APPLICATIONS

3.2.1 General Application Requirements

3.2.1.1 OS Services.

The rationale on what OS services can be used by Application are described in section 3.1.1 Operating System and its subsections.

The rationale for using CF File interfaces instead of OS file services is described in section 3.1.1.4.4.1.4 POSIX_FILE_SYSTEM Functions.

Standard termination of applications promote portability of waveform.

3.2.1.2 CORBA Services.

Standard set of CORBA Services promotes portability of waveform against Application and ApplicationFactory implementations.

The use of CORBA NAMING Service or stringified IORs options is based upon the Software Assembly Descriptor which is derived from the CORBA Components Definition.

3.2.1.3 CF Interfaces.

Standard interfaces allows for consistent behavior for Application and ApplicationFactory implementations.

Standard execute parameters (e.g., Naming Context) and format (argc, argv) promote portability of waveform.

Standard interchange format (XML) promotes portability of waveform and for consistent behavior for Application and ApplicationFactory implementations.

3.2.2 Application Interfaces

3.2.2.1 Utility Applications

3.2.2.2 Service APIs

3.2.2.2.1 Service Definitions

3.2.2.2.1.1 Format

All SCA-compliant Service Definitions shall have their interfaces described in IDL, except as allowed in 3.2.2.2.1.5.

It is required that all service definitions interfaces be defined using Interface Description Language (IDL). If it is expected that published APIs will be reused by many different vendors, it becomes important to have a standard interface description language. IDL was chosen because it provides a language independent language and because it can be directly compiled into CORBA ORBs and it fosters inheritance.

SCA-compliant Service Definitions shall conform to the Service Definition Description (SDD) provided in Appendix E, except as allowed in 3.2.2.2.1.5.

The SDD was developed to provide a single standardized language that could be used when publishing service definitions. A single standardized language will provide for consistency that will allow implementers to quickly access in formation from one SDD to another. The information required in the SDD was modeled after commercial standards and extracts the needed information about the interface for easy access by implementers. There is no requirement that a Service Definition description be limited to the information included in the SDD. It must include that portion defined in the SDD as a minimum.

3.2.2.2.1.2 Service Definition Usage and Creation

There are no requirements levied in this section. The following is provided for clarification:

The steps provided in this section are given in the order of precedence to promote common interfaces between protocol layers. The most advantageous method is to reuse an interface verbatim. The second most advantageous method is to inherit pieces of existing interfaces at the same layer to promote common functions among protocol entities. Steps 3 and 4 require starting a new inheritance tree which is not as advantageous as steps 1 and 2, but is required to incorporate future waveforms that are very different from existing waveforms and import commercial Service Definitions.

3.2.2.2.1.3 Service Definition Identification

Each Service Definition shall be identified by a UUID as defined in Section 7.4.

The association of a Universal Unique ID (UUID) to each service definition will aid in the tracking and management of the implemented interfaces. The UUID can also be used by the Domain Manager to identify which APIs are supported by a particular JTRS resource.

The requirements for interoperability and future plug-and-play require service definitions interfaces to be uniquely and unambiguously identified to ensure that different software objects communicate using an agreed-upon interface definition. The Domain Manager can use such an interface identifier as one of the factors in determining which software objects to instantiate together to form a protocol stack. (For example, if an IP network protocol could identify which Link Networking API Service Definition(s) it supported, the Domain Manager could use this identifier to consider only those LinkResources that supported the identified Link Networking API Service Definition.) Thus, there is a requirement for a unique and unambiguous way to identify Service APIs.

The Universally Unique Identifier (UUID) used in CORBA (similar to the Globally Unique Identifier used in COM/DCOM) provides a mechanism to uniquely and unambiguously identify system elements (such as objects and interfaces). The UUID is a 128-bit (16-byte) number that is algorithmically created using a machine's unique network card ID, the current time, and other data. No centralized authority is required to administer UUIDs (beyond the one that allocates IEEE 802.1 node identifiers [MAC addresses]). Because the UUID was mandated as standards based, CORBA compatible way to support the requirement for unique and unambiguous identification of Service APIs.

3.2.2.2.1.4 Registration

Public (i.e. non-proprietary) service interfaces used in SCA-compliant systems shall have their public Service Definitions and associated UUIDs registered as defined in Section 7.4.

Registration of a service definition, with a controlling body will allow for a single point for the collection and distribution of this information. Registered service definitions will provide an "official" standard for any waveform implementers interested in adopting a published service definition. In addition to be uniquely and unambiguously identified, Service API definitions must be made available if waveform implementers are to develop software that is interoperable and portable. In particular, any Service API definitions which are to be considered "public" must be made available to all vendors. A registration body provides a convenient and single location for vendors to register their public interface definitions and associated UUIDs. Other waveform implementers can then go to the registration body repository and re-use previously developed interfaces, thus increasing potential interoperability and plug-and-play and minimizes the effort required to develop new interfaces.

The registration body could encourage standardization and interoperability by categorizing the interfaces published for different waveforms as mandatory, recommended, allowable, experimental, etc. For example, four interfaces could be publicly registered for a SINCGARS-waveform Modem Networking API. The registration body could declare that one of these interfaces (say the simplest) is mandatory, another (say one that adds extensions to reduce co-site interference) is recommended, and the other two are allowable or experimental. SINCGARS-waveform implementers now know that supporting the mandatory interface will ensure that their software is compatible with other software SINCGARS-waveforms. In addition, if the implementers want to expend extra effort for improved performance, they can also implement the interface that supports improved co-site interference mitigation. Vendors are still free, however, to extend existing interfaces (or even develop new interfaces) to differentiate their product offerings from other vendors.

If the registration body categorizes the published interfaces, then it should be an open standards body similar to the IEEE, SDRF, IETF, etc. in order to reduce any unfair competitive advantage/bias that a single company or private consortium would have. Note that multiple registration bodies may be established. For example, there may be one registration body for military unique waveforms, one registration body for PCS waveforms, and one registration body for wireless LAN waveforms.

Private (i.e. proprietary) service interfaces used in SCA-compliant systems shall have their private Service Definition UUIDs, with a public description of the API, registered. They are allowed, but not required, to register their private Service Definitions with the Registration Body. [Note: it is a requirement for JTRS implementations of the SCAS that Service Definitions be public, as described in the API Supplement to the SCAS Specification.]

Registration, of private service definitions, allows for a comprehensive listing of what has been developed for the JTRS environment. It also provides an interested party with a point of contact if there is an interest in pursuing a licensing agreement with the developer of the private service definition. Configuration control of a software programmable radio is improved through the use of UUIDs that identify the service APIs supported by software resources.

3.2.2.2.1.5 Existing Service Definitions

There are no requirements levied in this section. No further explanation required.

3.2.2.2.2   API Transfer Mechanisms

3.2.2.2.2.1 Standard Networking Transfer Mechanism

The standard networking transfer mechanism shall be CORBA except as allowed in Section 3.2.2.2.2.2.

The JTRS Step 1 ADR described the over-the-air networking components as existing as applications with IDL interfaces and CORBA middleware as the transfer mechanism between them. Concerns over performance and commercial acceptability of such an architecture were expressed in the JTRS Networking Working Group. The JTRS Networking Working group undertook a system engineering analysis effort to determine whether the networking architecture, described in the ADR, was a valid one or whether some other architecture would be more appropriate. The analysis and subsequent scoring of the transfer mechanism options against criteria derived from ORD requirements essentially reaffirmed the architecture defined in the ADR. Portability was the differentiating factor in the analysis and therefore CORBA was established as the standard transfer mechanism for Networking APIs. The Architecture IPT later adopted the Networking Working Group's approach to services for their APIs for all SCAS services. The requirement for CORBA as the standard transfer mechanism was then generalized to include all services their APIs.

3.2.2.2.2.2 Alternate Networking Transfer Mechanism

An alternate networking transfer mechanism is allowed for the following case.

1. Application performance cannot be achieved with the standard transfer mechanism.

When an alternate transfer mechanism is used for real-time control and data flow, the transfer mechanism for initialization and non-real-time control shall use the standard transfer mechanism (if those controls can be separated).

The initialization and non real-time control and status in most cases, will be sent/received by an HCI either directly or indirectly through the CF. To retain location transparency between the controlled and controlling entities the interface to the controlled entity must use CORBA as the transfer mechanism.

When an alternate transfer mechanism is used, the transfer and message syntax of the alternate transfer mechanism shall be mapped to the IDL of the API Service Definition.

Mapping the transfer and message syntax of the alternate transfer mechanism to an existing API service definition allows reuse of large portions of the existing definition (behavior, state, etc). It also provides a translation from the alternate transfer mechanism interface to the IDL of the API Service Definition for the purpose of creating an adapter.

This mapping shall be identified by a UUID (separate from the Service Definition UUID).

Just as with the service definition, a mapping of an alternate transfer mechanism to a service definition is required. The requirements for interoperability and future plug-and-play require that this mapping be uniquely and unambiguously identified to ensure that different software objects communicate using an agreed-upon interface definition. Portability requires interfaces (whether publicly or privately documented) to be uniquely and unambiguously identified to ensure that a software object ported from one SCAS Radio System to another communicates with a software object that supports the desired interface rather than another object that supports an identically identified but different interface. For further information refer to sections 3.2.2.2.1.3 and 3.2.2.2.1.4.

The description of the alternate transfer mechanism, an analysis supporting the performance need for the alternate mechanism, the mappings to the Service Definition, and the associated UUIDs shall be registered as defined in section 7.4.

To maintain a high degree of interoperability and portability with the transfer mechanism implemented in different JTRS domains, it is required that CORBA be used. It is understood that there will likely be wideband waveforms that will have throughput requirements that may not be supportable by a CORBA transport. To support the cases where it can be shown through analysis that the CORBA transport would not meet the data throughput, alternate transfer mechanisms are allowed. The requirement to provide analysis forces the waveform implementers to analyze the requirements and the capabilities. The requirement to publish this information provides other waveform implementers with insight into cases where the standard mechanism was not adequate and gives the Government understanding and justification for the deviation from the standard transfer mechanism.

How the alternate mechanism maps to the Service Definition and the UUID that identifies this mapping aids in the tracking and management of the implemented interfaces. It supports interoperability and compatibility.

3.2.2.2.2.2.1 API Instance Behavior

Irrespective of the transfer mechanism used, all behavior including state transitions and priorities defined in the service definition shall be obeyed by an API Instance.

Without requiring that an API instance strictly conform to service definition there is no way to guarantee compatibility and interoperability at the interface. This requirement seeks to insure that the required components of the interface defined by the developer, are obeyed to provide for compatibility and interoperability.

3.2.2.2.2.2.2 Alternate Transfer Mechanism Standards

Transfer mechanisms shall be in accordance with commercial or government standards.

It is in the best interest of the JTRS community to define, adopt, and foster the use of commercial and government standards when developing radio components. Conformance to an open standard is an important criteria for any proposed transfer method that should not be waivered without strong technical rationale. The use of standard transfer mechanism(s) also results in lower acquisition and maintenance costs because the standard has already been developed and used thus benefiting from any economies of scale that have taken effect.

3.2.2.2.2.2.3 Alternate Transfer Mechanism Selection

There are no requirements levied in this section.

This section does describe goals of, considering the availability of supporting products (that have wide usage), are available from multiple venders, and are expected to have long-term support in the industry. Transfer mechanisms possessing these attributes can have a lower overall cost of ownership since they tend to have lower acquisition and maintenance costs.

3.3 GENERAL SOFTWARE RULES

3.3.1 Software Development Languages

3.3.1.1 New Software

The goal of new development is to produce SW that is abstracted from the details of the HW platform and Operating environment. Abstracting the details of the HW reduces/eliminates portability issues when changing processors, memory maps, bus protocols, etc. This does not guarantee 100% portable applications but it does greatly reduce the cost in porting SW to new platforms.

There are, however, applications that require performance that lower level languages provide. The use of lower level languages is allowed for special purpose functionality that has requirements that cannot be met by a High Order Language. Use of Low Level languages should be minimized to reduce its impact to portability and reuse. Interfaces to functionality provided by Low level languages should be as abstract as possible to reduce the scope of “less portable” SW.

3.3.1.2 Legacy Software

It is recognized that the use of legacy SW can have significant cost and or schedule benefits for some applications. When legacy SW is to be ported to a JTRS architecture, a Higher Order Language is not mandatory. It is necessary, however, that the legacy SW be “wrapped” within a SW interface that is compatible with the Core Framework. This is necessary to facilitate deployment of Legacy resources into the system and to provide for proper interface connectivity through CORBA. The Legacy application thus appears to the outside world as a standard JTRS resource, while retaining the Legacy implementation. This helps greatly to reduce the impact of porting Legacy SW.

4 HARDWARE ARCHITECTURE DEFINITION

No further explanation required.

4.1 BASIC APPROACH

See section 4.2

4.2 CLASS STRUCTURE

Attributes

Attributes can be viewed from two viewpoints within the hardware domain:

1. Generic HW module descriptive information.
2. Specific HW module descriptive information utilized by the radio Core Framework and/or software applications.

In the development of the hardware section of the SCAS, it was the finding of the Hardware Working Group (HW WG) that attributes shown in the SCAS could be approached only as representative lists. It was recognized that the true description of any hardware object / device / module is its documented specification. It was further recognized that any simple listing of module attributes could never fully describe a HW module's characteristics. Within the current framework of the HW architecture, the module specification will continue to serve the purpose of fully documenting the mechanical, electrical, and behavioral characteristics.

Attribute lists shown in the HW class diagrams in Section 4 of the SCAS represent a starting point for module specifications and potential use in Device Profiles. The actual attribute set for any HW module specification will be a function of specific radio requirements. The actual attributes reported to the Domain Manager through the Device Profiles will be a function of the software applications requiring HW information for operation. It is expected that module specifications will be essentially time-invariant during the lifetime of the hardware, but it is possible that the Device Profiles can be updated over the lifetime of the hardware as more waveforms and more sophisticated software applications are installed in the radios.

Class Structure

As a counterpart of the graphical representation of software, hardware is also depicted in the Object-Oriented UML format. When Object Orientation is looked at closely from the hardware view, it can be seen that it essentially describes the way that hardware has always been approached (i.e., partitioned into circuit cards and modules), but maybe not optimally.

The hardware class structure portrayed in the SCAS is provided to the radio designer as a template for the radio design process. It breaks the radio into small building blocks that are functional in nature. These functional building blocks are the hardware classes and subclasses. Figure 4-1 shows a simplified design process that the radio designer might use in designing a JTRS radio. Using this approach as a radio is designed, functional/performance allocation is first performed to the functional entities without regard to the physical nature of the ultimate hardware modules. As the allocations are performed, the attributes of the classes are assigned values. This mitigates premature association of function or performance to any particular hardware module. Once the functional and performance allocations have been fully performed, the radio designer can then proceed to the physical partitioning step to determine which functions will coexist within any given physical module. This facilitates optimal functional partitioning to physical modules based on the physical/ mechanical/ environmental requirements of the radio as compared to available hardware circuitry. This implies size/cost/performance/approach trades including the potential utilization of COTS/GOTS hardware modules.

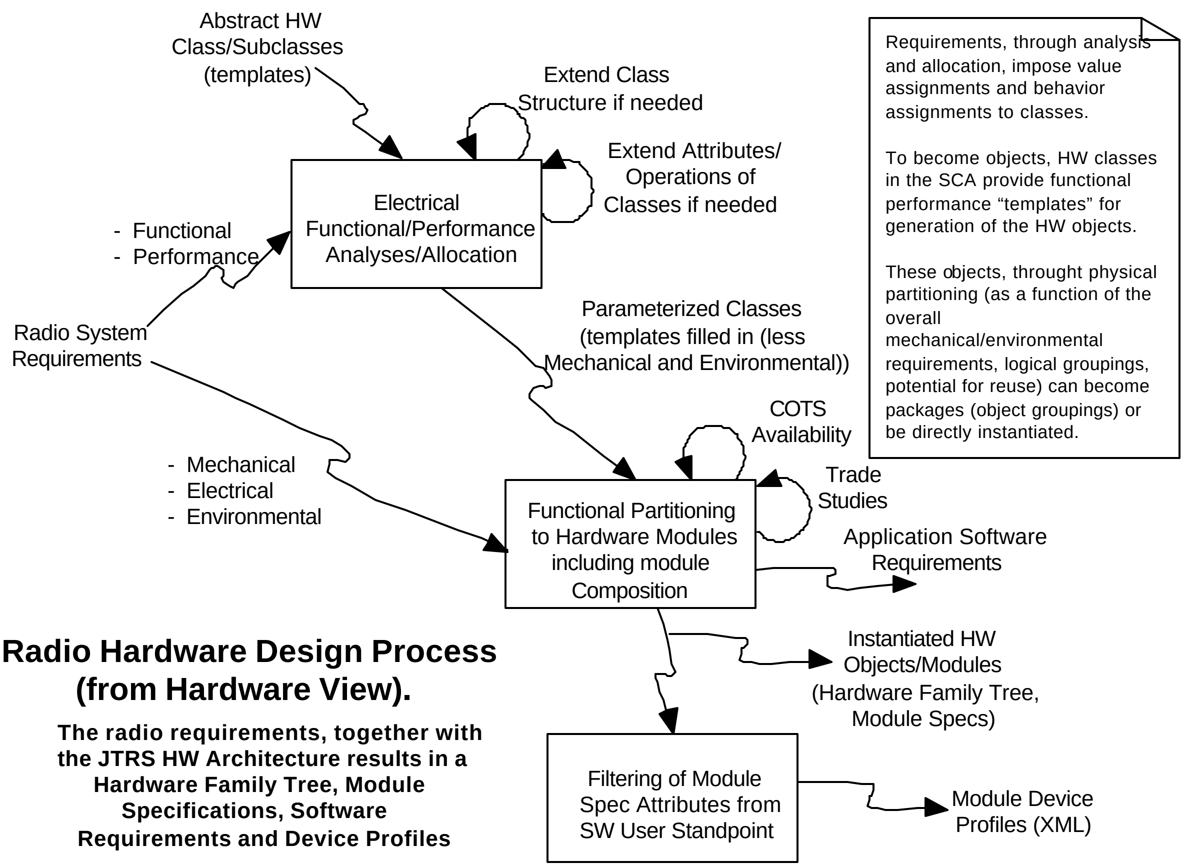

Figure 4-1. Radio Hardware Design Process

4.2.1 Top Level Class Structure

Stereotypes were utilized to generalize (or abstract) attributes in much the same way that higher level class structures generalize lower level classes. This was considered to be a key concept in understanding the role and relevance of attributes in the Object Oriented approach to hardware. In the HW class structure of the SCAS, potential software users, or "customers", of the attributes use stereotypes to sort or classify attributes. The abstraction of attributes into logical groups helps with the understanding and efficient tracking of attributes.

Also see section 4.2.

4.2.2 *HWModule(s)* Class Structure

Programmability

In SCAS draft versions 0.2 through 0.4, the concept of programmability was addressed by creating programmable and non-programmable classes. The programmable class represented hardware that had no functionality other than that provided by software running on the programmable hardware that was loaded by system software. The non-programmable class represented hardware that, due to its circuitry (i.e. dedicated hardware devices), had embedded functionality.

The shortcoming of this approach was quickly realized when it became apparent that all (or most) basic class concepts will produce programmable and non-programmable versions of the class. In order to stop the arbitrary proliferation of classes, the concept of programmability was captured as an attribute set or stereotype of the top level HW Module(s) class. All functional module classes could be realized as programmable, or non-programmable, as required. Multiple hardware functionality composed in a single hardware module can have the "programmable" attribute of each function, or sub-function, set as either programmable, or non-programmable, without the need for creating a new class.

The rationale, employed in developing the present approach for programmability also addresses the inherent programmable nature of a hardware object. Module developers can make the programmable nature of a particular hardware object available for use by the system software by proclaiming it with the "programmable" attribute. Hardware that is programmable obtains all or part of its functionality from software loaded onto it. On the other hand, a developer may develop an embedded function. A function realized by a software application running in a processor, is not provided from outside the module through system interfaces. For this latter case the programmability attribute may be set as non-programmable. In this case, the hardware class is not programmable, and its functionality is that of the embedded software and/or hardware.

Extensibility

The class structure provides a structured manner of grouping similar kinds of hardware together, recognizing that similar types of hardware generally have similar kinds of attributes. In this manner, like-hardware device properties can be compared and managed. In a similar manner, stereotypes are included to group attributes according to usage by user applications.

In developing the hardware architecture, it was realized that the notions of extensibility / expandability / extendibility must be an inherent part of the architecture. The architecture must provide freedom for new developments, while not placing a bound on creativity.

The hardware class structure in the SCAS is considered representative of current radio systems, but it is not meant to be all-inclusive for all implementations, nor to limit the description of hardware that does not fit into this structure. The addition of new technology may be represented by additional hardware classes. Also, information not adequately represented may be described by the addition of new or different attributes to existing classes. Attributes associated with applications not presently described may be grouped by the addition of new stereotypes.

GPS

The requirement of JTRS support for the GPS waveform is clearly identified in paragraphs 4.a.(2).(f) and 4.a.(2).(n) of the ORD for JTRS. For consistency with the OO approach to hardware and the implementation of waveforms, it was thought that GPS should be treated as any other waveform, with the exception that it is required for all domains, all platforms. However, early in the development of the hardware section of the SCAS, it was apparent that the hardware implementation for the GPS waveform using a mix of hardware devices, as for other waveforms, was not currently cost effective.

With the advance of technology, GPS receivers in the military and commercial markets have progressed to the point that COTS/GOTS hardware is extremely compact and relatively inexpensive. Given the cost advantage of COTS/GOTS hardware and the JTRS architecture's ability to accommodate legacy hardware, it was decided to represent GPS as a separate and distinct hardware module class (see Figure 4-2 in the SCAS). However, if compelling reasons point to the implementation of the GPS waveform using a mix of hardware devices, rather than an embedded and dedicated legacy receiver, the SCAS provides the flexibility for such an approach.

4.2.3 Class Structure with Extensions

No further explanation required.

4.2.3.1 *RF* Class Extension

No further explanation required.

4.2.3.2 *Modem* Class Extension

No further explanation required.

4.2.3.3 *Processor* Class Extension

No further explanation required.

4.2.3.4 *INFOSEC* Class

No further explanation required.

4.2.3.5 *I/O* Class Extension

No further explanation required.

4.2.4 Attribute Composition

No further explanation required.

4.3 DOMAIN CRITERIA

No further explanation required.

4.4 PERFORMANCE RELATED ISSUES

The architecture must allow for the arbitration and mitigation of potential interference pertaining to the use of the radio frequency spectrum for each included channel. Not only should the radio system address interactions between assigned channels it should be flexible enough to account for, and be tailored to, platform conditions. The architecture needs to address this coupling of real world installed environmental conditions into radio system performance. To accomplish this, the architecture should support incorporation of dedicated cosite mitigation hardware either self contained in the radio chassis, or externally within an ancillary unit that will be referred to as an Antenna Distribution Unit (ADU). Additionally the architecture must allow for the incorporation of a software cosite manager function that can utilize knowledge of the radio (hardware), waveform (software) and platform (installation) parameters to perform real time prediction of potential cosite interactions.

These goals are accomplished within the hardware architecture through the utilization of the flexibility afforded with the JTRS OO approach. Specifically, within the hardware architecture, cosite has been addressed at both the class/subclass and attribute level. Figure 4-2 shows representative subclass and attribute relationships. The RF, Modem and I/O classes, subclasses

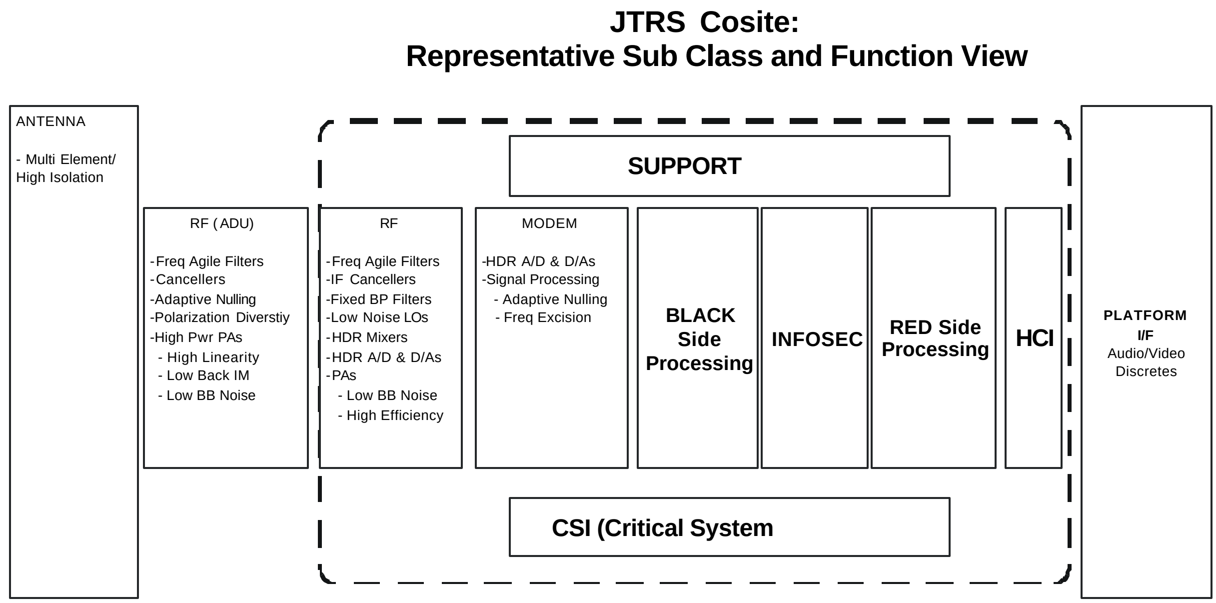

Figure 4-2. Cosite Subclasses and Attributes

and attributes support the definition and substantiation of the cosite mitigation hardware and its associated performance. To further help define the cosite interactions between classes a cosite stereotype <<CositePerformance>> has been created in the RF top class and is inherited by all RF subclasses, or class extensions. Additional detail for the cosite stereotype has been added to the Receiver, Exciter and Power Amplifier subclasses (see SCAS, Figure 4-3, RF Class Extensions). Finally, recognizing the strong dependency of dedicated cosite mitigation devices to the RF arena, a separate cosite subclass was added to the RF class. This object-oriented approach enables a consistent application of the HW architecture (classes and rules) across the various domains (i.e., Handheld, Dismounted, Vehicular, Airborne, and Maritime/Fixed) and will allow for a JTRS installation into platforms with anything from a mild to severe cosite environment. Examples of how cosite solutions can be realized within the JTRS framework are provided below.

Simultaneous contention for frequency spectrum by multiple transmitters and receivers must be eliminated. Degradation is usually inversely related to frequency separation, and may be mitigated by several methods that are supported by the SCAS.

The SCAS supports specialized hardware that is often employed to decrease cosite related degradation. This is supported via an RF subclass for cosite mitigation hardware. SCAS also supports software techniques, such as cosite manager, via attributes useful to such an application. A cosite manager application can authorize emissions or command protective measures based on instantaneous conditions. Protective measures, (for example inserting attenuation, blanking a signal, or overriding AGC), while usually degrading, are usually preferred to uncontrolled conflicts. Other protective measures, such as time slot scheduling, may not degrade operation.

Examples of cosite mitigation hardware are preselectors, postselectors, cross-band filters, multicouplers, and signal canceller devices. The performance of all such devices can be characterized by the properties of the attenuation and bandwidth attributes. For example, the system performance of signal cancellation devices can be described using these attributes to report cancellation depth (attenuation) and bandwidth. Similarly, filter characteristics are defined by values reported by these attributes. Additional information, such as tuning speed, and frequency, relevant to cosite mitigation devices is already available as attributes "inherited" from the general RF class.

A Cosite Manager application is supported by the SCAS to provide software control of assets to mitigate frequency contention. The Cosite Manager must first recognize potential conflicts. In order to determine this, attributes (identified by the <<CositePerformance>> stereotype) provide source information / data. Examples of cosite performance related attributes are transmit wideband noise, and receiver (or LNA) dynamic range (likely reported via a third order intercept or similar method).

Note that the users of attributes are not mutually exclusive. The cosite manager will also use other attributes, such as <<performance>> stereotyped attributes. Examples of these are PowerLevel, NoiseFigure, etc. Some cosite performance attributes such as dynamic range are also performance related. Information and data of software performance (for example DSP) and platform performance (for example antenna coupling) will also be needed by the cosite application in order to compute potential degradation and command mitigation.

4.5 GENERAL HARDWARE RULES

No further explanation required.

4.5.1 Device Profile

The use of attributes to describe hardware is the heart of the object-oriented approach of the SCAS. The entire hardware class structure is based on grouping these attributes according to device types (classes) and on methods of organizing them (stereotypes) according to application users.

The Device Profile is used to collect and report needed attributes and their values to the system software. The Device Profile for hardware devices consists of the Device Component Descriptor

File and the Device Descriptor File. The Device Profile and its files are detailed in SCAS 3.1.3.4. XML is the language chosen for the Device Profile as described in SCAS 1.3.1.3.

Attributes within the Device Profile are those attributes needed by software applications. The applications can be the Domain Manager, or other applications such as waveform software. Some applications, such as a Cosite Manager (see section 4.4), may require more than single valued attributes, so these attributes may be presented as data files or links to data files.

4.5.2 Hardware Critical Interfaces

Early in the hardware structure definition, the hardware working group recognized that critical hardware interfaces should be a function of the point of sale. In other words, if you are buying a radio, then the critical interfaces are the radio box inputs and outputs. If you are buying modules, then the critical interfaces are the module inputs and outputs. Recognizing that the effective point of sale for JTRS radios is both at the radio, and module level, (since JTRS radios are required to be modular per the ORD), the definition of critical interface must take the more detailed view (module level). Thus, the SCAS defines all inputs and outputs from hardware modules to be critical interfaces. This provides the benefits of allowing competitive procurement of HW modules as well as making HW modules available for reuse in other radios.

Recognizing that defining critical interfaces at this level provided benefits, it was further determined that there was no additional benefit from requiring interfaces to be defined in any more detail (i.e., internal interfaces within a module). Defining critical interfaces at the module boundaries and not inside module boundaries, helps to protect hardware developers from having to share proprietary rights/intellectual property in order to sell radios and/or modules. In other words, one company could develop a module with certain (critical) interfaces and performance attributes using discrete circuitry. Later, another company could supply a module to the same exact specification (external interfaces and performance) using a proprietary integrated circuit. The interfaces internal to the module, in either form would not need to be revealed in detail. This is not meant to prevent a supplier from making internal interfaces available at module boundaries either for other potential radio uses or for proprietary programming purposes.

Interfaces at module boundaries are required to be in accordance with commercial or government standards to foster use of COTS/GOTS hardware and further to foster reuse of newly developed modules. This should also make it easier and more attractive for commercial hardware suppliers to become hardware suppliers to military radios.

Unique interfaces are permissible when warranted by radio performance requirements. This might include overload of the selected standard bus structure or other specific requirements that a radio must meet. By requiring these exceptional interfaces to be clearly and openly defined still maintains the benefits of competitive procurement and reuse, while eliminating technological roadblocks in radio developments.

4.5.2.1 Interface Definition

See section 4.5.2.

4.5.2.2 Interface Standards

See section 4.5.2.

4.5.2.2.1 Interface Selection

See section 4.5.2.

4.5.3 Form Factor

Commercial standards provide the maximum ability to re-use existing developments, provide technology insertion, and leverage COTS technology growth as much as possible. This is not possible with proprietary standards that impede open system architectures. At the implementation level (below the architecture level of the SCA), defining and encouraging specific form factors provides the maximum interchangeability and reuse of modules. However, no single form factor is applicable across all domains. Further, no form factor can be totally standardized within a given implementation. Certain unique items like power supplies and reference standards may not be conducive or may not fit in a chosen standard. The ORD recognizes this when it states that the JTRS system "shall provide an internal growth capability through an open systems architecture approach … and shall be modular, scaleable, and flexible in form factor[3]." Using a "should" rather than a "shall" in SCAS section 4.5.3 is due to the need to maintain flexibility in the form factor.

4.5.4 Modularity

In tracking compliance with the Joint Technical Architecture, the concept of modularity has been established as a requirement in the ORD for the JTRS. Though the concept of modularity is common in the development and construction of today's electronic systems, modularity has been directly addressed as a threshold Key Performance Parameter (KPP) requirement in paragraph 4.a.(1).(b)) of the ORD. In doing so, the ORD has emphasized scalability and flexibility in the form factor, which has directly influenced hardware rules in the SCAS.

---

[3] Operational Requirements Document for Joint Tactical Radio, 23 March, 1998, (1) General Performance Requirements (a)

5 SECURITY ARCHITECTURE DEFINITION

No changes or additions have been incorporated into the Security sections of the SCAS or SRD for release 1.1. Currently on-going Security Working Group activities will result in new material SCAS/SRD release 2.0.

Security for the JTRS radio, not only encompasses the radio implementation but also the architecture as well. Traditionally, security analysis has been relegated to the physical domain. This section of the SRD discusses rationale for various security requirements that have been included in the SCAS and the companion document, the Security Supplement. The architectural requirements listed are an abstraction of requirement from the UNIFIED INFOSEC CRITERIA, UIC; which contains both architectural and implementation details.

The requirements are organized into three major categories: cryptographic (Crypto), non-cryptographic (NC) and system. Requirements overlapping categories were placed in a ‘best fit’ category.

The cryptographic category includes those architectural requirements that affect the COMSEC subsystem. This category includes specific cryptographic functions and the support required to execute those functions. The non-cryptographic category includes those architectural requirements that are within the INFOSEC boundary. The INFOSEC boundary includes the COMSEC subsystem and the controls that directly affect the COMSEC subsystem. The system category includes those architectural requirements that have security implications from a radio perspective. This category includes HCI, BIT, power, etc.

The requirements discussion includes a requirements description, whether the requirement is architecture or implementation, where it fits in the architecture and what implied requirements exist.

Of the three major categories, the user application functions within the JTRS radio have been further divided into six categories as follows.

- Crypto Functions
- Crypto Support Functions
- NC Security Functions
- NC Security Support
- System Application Functions
- System Functions

These six categories are addressed, along with their subcategories, in terms of the following issues.

- Is it architecture and/or implementation?
- Where does it fit within the JTRS architecture?
- What it's implied requirements are?
- What is the description of the category/subcategory?

In some cases, the formulation of the details is still in process, but the functions, without detail, are listed for completeness.

5.1 CRYPTO FUNCTIONS

5.1.1 Encrypt/Decrypt

Description: Encryption is a means of enciphering and encoding data for secure transmission. Decryption is the reverse roll of encryption, were the data is decoded and deciphered.

Architecture/Implementation: The architecture allocates this function to the INFOSEC module. However, the actual encryption/decryption process is an implementation issue based on the Crypto algorithm being performed.

Place in the Architecture: INFOSEC Module

Implied Requirements:

- Data for encryption must be isolated from Bypass data
- The actual encryption is a Implementation that is performed by the INFOSEC Module

5.1.2 Crypto Alarm

Description: Equipment Alarms involve monitoring specific components of the system for evidence of failures that could result in security problems. The monitoring of these alarms and their indicators must be designed into the system.

Architecture/Implementation: Alarm functions become part of the Architecture within the INFOSEC Module and are also part of the implementation of the system. The Alarm indicators become part of the architecture along with the action taken when an Alarm occurs. The actual communication between modules within the system is part of the system implementation.

Place in the Architecture: INFOSEC Module and HCI

Implied Requirements:

- Report Alarm condition to the operator.
- Provide an alarm indicator on the front panel of the terminal.
- Shut down all cryptographic function until alarm condition is resolved.
- The implementation must have the hooks for data transmittal of the alarm condition throughout the system.
- Shall support isolation of an Alarm to a channel when appropriate.
- Inhibits all cipher text output on channel in Alarm
- Shut down all channels if a catastrophic Alarm occurs

5.1.3 Zeroize All

Description: Removal or elimination of keys from a Crypto equipment or storage.

Architecture/Implementation: Architecture, The architecture must be capable of stimulation of the Zeroize command for both internal and external commands. Zeroization of keys can be performed in several manners, which effect the architecture.

Place in the Architecture: INFOSEC Module and HCI

Implied Requirements:

- Activation of Zeroize process by an active signal.
- Zeroization of Keys do to a TAMPER detect signal going active.
- Zeroization of Keys do to a non-scheduled power loss.

5.1.4 Selective Zeroization

Description: Removal or elimination of a key or keys from a Crypto equipment or storage.

Architecture/Implementation: Architecture, The architecture must be capable of stimulation of the Zeroize command for both internal and external commands. Zeroization of keys can be performed in several manners, which effect the architecture.

Place in the Architecture: INFOSEC Module and HCI

Implied Requirements:

- Activation of Zeroize process by an active signal.
- Selective zeroization of keys
- Activation of zeroization process by an OTAZ message.

5.1.5 Erase Algorithm

Description: The erasure of a cryptographic algorithm from internal storage.

Architecture/Implementation:

Place in the Architecture:

Implied Requirements:

5.1.6 Programmable Crypto

Description: It is a set of hardware and software capable of changing COMSEC algorithms and keys (Need unwrap key) in a tactical environment. This allows the device to interoperate with several different COMSEC devices.

Architecture/Implementation: Architecture pertaining to the INFOSEC Module, The architecture needs to supply the I/O required for interfacing to the different possible programmable Crypto devices.

Place in the Architecture: INFOSEC Module

Implied Requirements:

- Provide secure storage of Crypto algorithms.
- Provide secure Key storage.
- Provide interoperability with legacy waveforms.

- Must conform to all FSRS requirements which legacy Crypto fulfilled.
- Association of Algorithm to information user

5.1.7 External Cryptography Support

Description: A piece of cryptography, which is external to the system. This external cryptography communicates with the system via I/O ports.

Architecture/Implementation: Architecture, The architecture must support the communication path to the external cryptography.

Place in the Architecture: HCI, INFOSEC Module

Implied Requirements:

5.1.8 Bypass Control, Data, and Protocol

Description: Cryptographic bypass is the function of routing information around the Crypto logic. The bypassed data could be control/status information, Plain text bypass of data, or protocol information.

Architecture/Implementation: Architecture, In the area of Plain text bypass a deliberate action by the operator must take place before plain text data can be bypassed. The bypass of control/status and protocol information falls on both the architecture and implementation. The architecture must provide a means of routing the information for bypass.

Place in the Architecture: INFOSEC Module

Implied Requirements:

- Provide a Bypass for communications protocol/Header based on information content.
- Provide a Plain Text Bypass that is only be invoked by an overt manual operation (transmit).
- Provide a local access control mechanism for Bypass
- Monitor Bypass channel

5.1.9 TRANSEC

Description: TRANSEC is a measure used to protect transmissions from interception and exploitation.

Architecture/Implementation: Both Architecture and Implementation. The architecture needs to be able to handle where the TRANSEC operation will be performed (i.e. RED side/ BLACK side). The actual performance of the TRANSEC function is an implementation issue when being performed either on the INFOSEC module or on the Modem.

Place in the Architecture: INFOSEC Module, BLACK side (Processor, Modem)

Implied Requirements:

- Support the execution of TRANSEC (i.e. Key load algorithm)

- Load time
- Maintain time (with or without power)
- Use TOD

5.1.10 Multi-Channel Operation

Description:

Architecture/Implementation:

Place in the Architecture:

Implied Requirements:

5.2 CRYPTO SUPPORT FUNCTIONS

5.2.1 Digital Signature (High Grade)

Description: Accept Block message (s) for signature validation.

Architecture/Implementation: Architecture pertaining to the INFOSEC Module

Place in the Architecture: INFOSEC Module

Implied Requirements:

- Shall be capable to select what kind of signature.
- Select PKI certification or Key.
- Submit file length.
- Result returned (meaning if digital signature was unwrapped)
- Get time
- Paths to support RED, BLACK & BLACK, RED

5.2.2 External Algorithm load

Description: Download of Crypto Algorithm from an external means.

Architecture/Implementation: Architecture in the support of DS101, user interfaces.

Place in the Architecture: HCI, INFOSEC Module.

Implied Requirements:

- Selection of the Algorithm.
- Associate the algorithm ID with the proper algorithm. (Binding, Tagging)
- Authentication
- Integrity check
- DSS, Hash, Decrypt (process) storage

5.2.3 Internal Algorithm load

Description: Loading of internally stored algorithms into the Crypto by ID.

Architecture/Implementation: Implementation.

Place in the Architecture: INFOSEC Module.

Implied Requirements:

- Decrypt DSS, Hash
- Identification
- Verification and test

5.2.4 BLACK External Load Key

Description: Key load is the method of loading key either from a key fill device or over the air.

Architecture/Implementation: Implementation.

Place in the Architecture: INFOSEC Module and HCI

Implied Requirements:

- Provide fill operations for a minimum of BLACK fill up to Benign fill capability
- Encrypt store
- Decrypt check
- Integrity

5.2.5 RED External Load Key

Description: Key load is the method of loading Key from either Key fill device (i.e. DTD, KYK-13, KOI-18…) or by an Over The Air Rekey Message.

Architecture/Implementation: Architecture, Keys will be loaded via two methods which are, direct fill and by an OTAR message. The direct fill will be done via a common fill port. The fill lines from the fill port will go directly to the INFOSEC Module. The INFOSEC Module will steer the keys directly to the key storage location (i.e. RAM Storage or internal storage to the COMSEC device).

Place in the Architecture: INFOSEC Module and HCI

Implied Requirements:

- Provide fill operations for a DS101/DS102 interface
- Provide capability for DS100 key tagging
- Provide capability for Multi key for different Algorithm
- Provide for OTAR capability for supporting waveforms
- Encrypt and store

5.2.6 Use Key

Description: The use of the key in performing the COMSEC and TRANSEC functions of the waveform.

Architecture/Implementation: Architecture, Interfaces requirement with the operator.

Place in the Architecture: HCI

Implied Requirements:

- Unique key ID's associated with each Key
- Validation of key
- Selected key
- Decrypt
- Lead outside Boundary (e.g. TRANSEC key)

5.2.7 External Zeroize

Description: The zeroization of key from an external means.

Architecture/Implementation: Architecture, The architecture must provide the means for handling an input from a user interface.

Place in the Architecture: HCI, INFOSEC Module

Implied Requirements:

- Provide means to respond to an external zero-all switch.
- Means of zeroization by the operator using a mode Zeroize switch.
- Means of zeroization by means of a Tamper detect and under/over voltage.

5.2.8 Synchronize/Resynchronize

Description: Algorithm dependent, bit alignment/Frame Alignment for Crypto sync.

Architecture/Implementation: Both Architecture and Implementation. The architecture needs to be able to handle where the Sync/Resync operation will be performed (i.e. RED side/BLACK side). The actual performance of the Sync/Resync is an implementation issue.

Place in the Architecture: INFOSEC Module, BLACK side (Processor, Modem).

Implied Requirements:

- Support execution of Sync/Resync.
- Time base
- Get timing
- Use Randomizer (Message Indicator based)

5.2.9 Sense Patterns

Description: These are message patterns contained within the message

Architecture/Implementation: Architecture, The Architecture needs to handle real time messaging from user I/O control or radio I/O control.

Place in the Architecture: RED/BLACK Side, HCI

Implied Requirements:

- Waveform dependent detective device which shall notify user
- Control performance of automated function (SICGARS, EOM, and OTAT…)

5.2.10 Time Of Day (TOD)

Description: Time tracking for cryptographic sync.

Architecture/Implementation: Both architecture and implementation. Architecture needs to carry the real time (e.g. GPS, Net controller) clock function. The actual communication of the TOD between modules within the system is part of the implementation.

Place in the Architecture: Modem, INFOSEC Module, RED/BLACK side of terminal.

Implied Requirements:

- Battery to real time clock
- Potential use of control Bypass
- Get time
- Provide or validate time stamp

5.2.11 Randomization

Description: Generation of random numbers for multiple purposes (i.e. CIK, Crypto randomization Patterns).

Architecture/Implementation: Implementation

Place in the Architecture: HCI, INFOSEC

Implied Requirements:

- User actions for timing

5.2.12 Control Functions

Description:

Architecture/Implementation:

Place in the Architecture:

Implied Requirements:

- Set Crypto video

- Command Bypass
- Zeroize

5.2.13 Over The Air Rekey (OTAR)

Description: A method of rekeying a user channel over the air.

Architecture/Implementation: Both Architecture and Implementation. Architecture must provide the means to detect the OTAR, either by the INFOSEC or the Modem. The implementation is the passing of the information between modules.

Place in the Architecture: INFOSEC Module, Modem

Implied Requirements:

- Notify user of OTAR receipt
- Assign key ID for received OTAR
- Key update
- Wrap key for storage

5.2.14 Over The Air Zeroization (OTAZ)

Description: A method of zeroization of key or keys by means of an over-the-air message.

Architecture/Implementation: Architecture, The architecture must supply the means for the detection of the OTAZ message by either the INFOSEC module or the Modem.

Place in the Architecture: INFOSEC Module, Modem

Implied Requirements:

- Pattern detection capability for the Modem and INFOSEC
- Authenticate command

5.2.15 Over The Air Transfer (OTAT)

Description: A key file transfer by means of over the air communication.

Architecture/Implementation: Implementation

Place in the Architecture: INFOSEC Module/key storage

Implied Requirements:

- Data storage
- User notification
- Message authentication/integrity
- KEK selection

5.2.16 HCI (Security Relevant Status Functional)

Description: The HCI shall be able to identify algorithm and key IDs contained within the system.

Architecture/Implementation: Both.

Place in the Architecture: HCI, Programmable Cryptography, Key Manager, and Domain Profile.

Implied Requirements:

- The system administrator shall be able to retrieve system configuration and status information.
- This information is needed to ascertain the system configuration information in preparation to mission planning.
- Security critical configuration information includes algorithm and key status.
- The algorithm and key status information shall uniquely identify each algorithm and key within the system.
- The system administrator is then able to identify algorithms and keys that may need to be loaded into the JTRS prior to the next mission.

5.3 NON-CRYPTOGRAPHIC SECURITY FUNCTIONS

5.3.1 Initialization

Description: Generation of Crypto/sync for encryption/decryption.

Architecture/Implementation: Architecture, In receive mode the Modem must have the capability to detect and send to INFOSEC Module.

Place in the Architecture: Modem/INFOSEC

Implied Requirements:

- Detection of sync patterns

5.3.2 Digital Signature II (Low Grade)

Description: A digital signature algorithm is used by a signatory to generate a digital signature on Hashed data and by a verifier to verify the authenticity of the signature. An asymmetric algorithm is used for digital signatures.

Architecture/Implementation: Both. The architectural issues are related to message structure support. Implementation is the primary issue.

Place in the Architecture: Red processing/Black processing. Architecture does not prevent you from handling in INFOSEC boundary.

Implied Requirements:

- Digital signatures shall be used to provide integrity and authentication mechanisms.
- The architecture shall support digital signature messages.

5.3.3 Authentication

Description: To establish the validity of a claimed identity. The establishment of identity may be provided by different mechanisms depending on the strength of mechanism required. Authentication may be as simple as using user-name and password pairs if weak mechanisms are all that is required. Stronger mechanisms can include public key/private key cryptography or Crypto ignition keys (CIKs). Hardware tokens, using public key/private key cryptography and passwords, may be used for stronger authentication, e.g. MISSI Fortezza cards.

Architecture/Implementation: Both. The architectural issues are related to message structure support. Implementation is the primary issue.

Place in the Architecture: Within the INFOSEC.

Implied Requirements:

- The authentication mechanism shall reside within an INFOSEC boundary in the JTRS.
- Accommodate table privileges, token interfaces
- Return results and execute

5.3.4 Privilege establishment

Description: Privileges refer to actions permitted by individual system entities.

Architecture/Implementation: The architecture supports the assignment of privileges for various entities in the JTRS. Implementation provides for the management of the privileges.

Place in the Architecture: The architecture shall provide the capability for the assignment of privileges to core applications, non-core applications, CORBA ORB entities, and I/O devices based on the security policy.

Implied Requirements:

- The architecture shall provide the capability for the assignment of privileges to various entities within the JTRS system. (Entities may be core applications, non-core applications, CORBA ORB entities, and I/O devices, system administrators, radio network users, or input/output ports.)
- The security policy shall identify the privileges of each system entity.

5.3.5 Data Separation

Description: Data separation is supported by the access control mechanism and access control database. The access control mechanism and access control database provide for secure message routing within the JTRS architecture.

Architecture/Implementation: Both. The architectural issues are in the existence of the access control mechanism and access control database. The implementation of these security functions is left to the developer.

Place in the Architecture: Access control mechanism and access control database.

Implied Requirements:

- No additional architectural impacts are required.

5.3.6 Application Separation

Description: Application separation shall be supported by the separation mechanism. The separation mechanism provides for process separation between applications.

Architecture/Implementation: Both. The architectural issues are in the existence of the separation mechanism. The implementation of this security function is left to the developer.

Place in the Architecture: Separation mechanism.

Implied Requirements:

- No additional architectural impacts are required.

5.3.7 Major Function: Access Control

Separation of data is a primary concept in computer security. In secure computers, this concept can take at least three forms: trusted computer base (TCB), access control mechanism, and separation mechanism. The TCB is the security-critical (trustworthy) part of the system; all other parts are presumed to be hostile (fraught with Trojan horses, for example).

The access control mechanism is a subpart of the TCB. The access control mechanism arbitrates access requests to the system resources and enforces those access decisions. The access decisions are based on a specified system security policy (often embodied in an access control list/database) and on the security attributes of the requestor and requested resource. See Figure 5-1 for a model of an access control mechanism implementation. The three design requirements that must be met by an access control mechanism are:

- The access control mechanism must be tamperproof;
- The access control mechanism must always be invoked;
- The access control mechanism must be small enough to be subject to analysis and tests, the completeness of which can be assured.

Figure 5-1 illustrates the implementation of an access control mechanism. The access-control database is the element of the system that embodies the security policy. The access control mechanism can be used to support different system security policies with changes to the access control database.

A separation mechanism is the subpart of the access control mechanism, which isolates system resources (memory, input/output devices, processing time, storage devices like disk, etc.) between various processes on the system. A common mechanism for implementing a separation mechanism is a memory management unit (with associated software). Thus, the access control mechanism provides for controlled sharing amongst the isolated domains created by the separation mechanism.

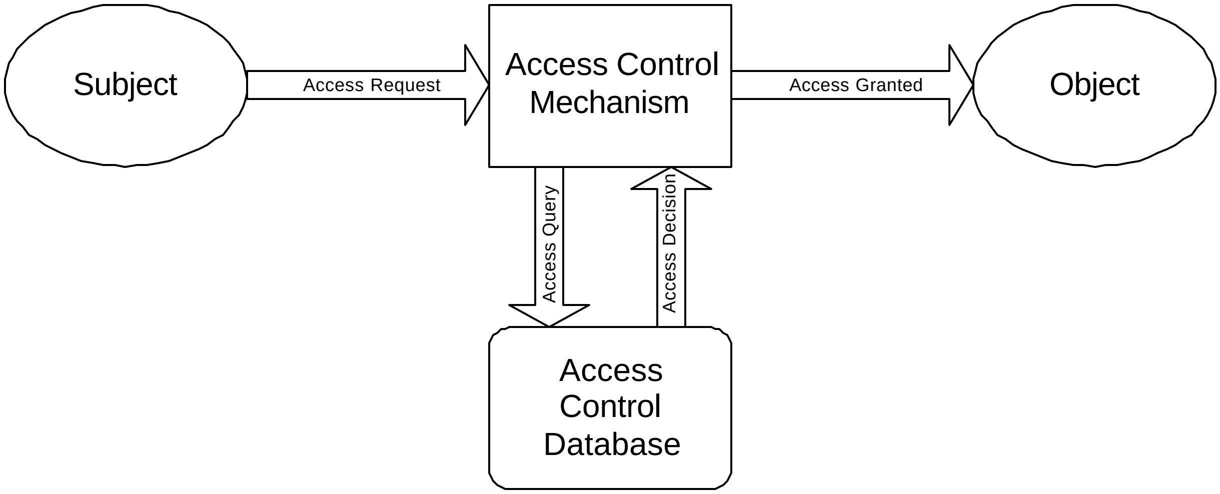

Figure 5-1. Generic Access Control Mechanism Model

5.3.7.1 <u>Sub-Function:</u> Access Control Mechanism

<u>Description</u>: The access control mechanism arbitrates access requests to the system resources and enforces those access decisions. The access decisions are based on a specified system security policy (often embodied in an access control list/database) and on the security attributes of the requestor and requested resource.

<u>Architecture/Implementation</u>: Both. Access control mechanism applies to the architecture in providing a framework for securely establishing non-core applications, and secure messaging within the architecture. The implementation will develop the appropriate mechanism(s) instantiating the access control mechanisms.

<u>Place in the Architecture</u>: Core framework, Message Registration, Control/Status Bypass, Protocol Bypass, and User (system administrator and potentially other users) access.

<u>Implied Requirements</u>:

- Core framework shall implement the access control mechanism portion of the reference monitor concept for system control functions.

5.3.7.2 <u>Sub-Function:</u> Separation Mechanism (Access Control)

<u>Description</u>: A separation mechanism is the subpart of the access control mechanism, which isolates system resources (memory, input/output devices, processing time, software processes, mass storage devices, etc.) between various processes on the system.

<u>Architecture/Implementation</u>: Both. The existence of a separation mechanism is architectural. The implementation of the separation mechanism is left to the implementor.

<u>Place in the Architecture</u>: A unique separation mechanism shall exist on all Red-side nodes and Black-side nodes as applicable (waveform dependent).

<u>Implied Requirements</u>:

- The architecture shall support a separation mechanism.

- Paths
- Processes
- Data

5.3.8 Audit

Description: Audit provides a mechanism for recording significant events. The audit log will be protected from deletion or even being read by unauthorized entities. Audit information will be provided to the system administrator and Security officer.

Architecture/Implementation: Both

The existence of audit is an architectural feature.

Place in the Architecture: Audit log functionality for JTRS shall be provided by the operating environment.

Implied Requirements:

- Determine audible events
- Each reportable process is responsible for reporting any audible events to the audit logger.
- Protect the audit log information from being tampered with or accessed by unauthorized entities.

5.3.9 Examples of Implementation (Firewall)

Description: The purpose of a firewall is to provide controlled and audited access to services, both from inside and outside an organization's network, by allowing, denying, and/or redirecting the flow of data through the firewall. Firewalls fall into two major categories: traffic-filter and application-level firewalls.

Architecture/Implementation: Implementation: Firewalls may be used to support network management applications or the implementation of user interfaces.

Place in the Architecture: Not applicable.

Implied Requirements: None.

5.3.10 Classified Application

Description: Classified applications refer to applications whose object code is classified.

Architecture/Implementation: The architecture shall be able to decrypt classified applications and securely route the classified object code to the appropriate process space.

Place in the Architecture: Security requirements for securely routing of classified object code shall be supported by the separation mechanism, access control mechanism, core framework, and secure message connectivity.

Implied Requirements:

- All classified object code shall be stored encrypted when not in use.
- The architecture shall support process isolation for classified processes.
- The architecture shall support message isolation for file transfer of classified applications.
- Erase Red application on certain applications (e.g. Declassified terminal)

5.3.11 Data Integrity

5.3.11.1 Sub-Function: Internal Data Integrity

Description: The state that exists when computerized data has not been exposed to accidental or malicious alteration. Internal data integrity provides assurance that messages within the JTRS have not been altered between client and server.

Architecture/Implementation: Both. The architecture shall support data integrity services. The integrity service implementation is left to the developer.

Place in the Architecture: Integrity mechanisms shall support message traffic within the architecture.

Implied Requirements:

- The operating environment shall support data integrity mechanism for message traffic within the JTRS architecture.
- Integrity for programs applications check (e.g. files, tables)

5.3.11.2 Sub-Function: External Data Integrity

Description: The state that exists when computerized data has not been exposed to accidental or malicious alteration. External data integrity provides assurance that messages received from outside the JTRS have not been altered from the trusted source. Data integrity can be implemented in conjunction with authentication services to provide secure distribution of data or software files.

Architecture/Implementation: Both. The architecture shall support MISSI integrity services. The implementation may use Type I or Type II integrity services. Data integrity services may be implemented in hardware, software, or tokens. Cryptographic algorithm integrity shall use an approved authentication service.

Place in the Architecture: Any integrity mechanism shall be accessible by the core framework. Cryptographic algorithm integrity shall be implemented within equipment authorized by the appropriate certification authority.

Implied Requirements:

- The operating environment shall support integrity services (e.g. digital signal secure hash)

5.4 NON-CRYPTO SECURITY SUPPORT FUNCTIONS

5.4.1 Security Policy

Description: Security policy provides for the secure operation of the JTRS.

Architecture/Implementation: Both. Architecturally, the security policy is embodied in the access control database. The policy will vary from program to program because of the different security policies of each program.

Place in the Architecture: Architecturally, the security policy is embodied in the access control database. Implementation of the access control database is left to the developer.

Implied Requirements:

- The security policy shall be embodied by an access-control database, which encapsulates the security policy in a way that system entities (e.g. access control mechanisms) may enforce the security policy.
- Security policies that are enforced at the system level shall be embodied in the Security policies.
- Security policies may also be enforced at a more local level. For example, the control/status bypass enforces the security policy that relates to the unencrypted system level information that crosses the Red/Black Boundary. A waveform may have a security policy as well. An example of a waveform security policy is when the housekeeping information crosses the Red/Black boundary over the protocol bypass.
- Load Security Policy

5.4.2 Access-Control Database (Security Policy)

Description: The access-control database is the element of the system that embodies the security policy. Different system security policies can be supported by changes to the access control database without modification of the access control mechanism.

The JTRS will use a "flow control" security policy. A flow control security policy is used because of its compatibility with the object oriented data model. A more traditional security model, such as the Bell and LaPadula model, uses the concept of passive objects and active subjects. The flow control model uses objects and messages. An object in the flow control model may possess properties of a passive data container, and an active agent. Messages become the method of transferring information between objects. The flow control model fits well with the object oriented design process and the information flow within the JTRS.

Architecture/Implementation: Both. The use of access control databases in conjunction with access control mechanisms is an architectural feature. The implementation of the access control database is left to the developer.

Place in the Architecture: The operating environment shall support and enforce the system security policy. Other security policies may exist at a more local level, such as the Control/Status Bypass.

Implied Requirements:

- The operating environment shall implement the access-control database for system control and message routing functions.
- The operating environment shall contain the necessary information that identifies the software and hardware resources necessary to establish non-core applications without violating the security policy.
- The operating environment shall contain the necessary information required to provide message registration (establish virtual communication channels) between those resources that will provide the non-core applications.
- The JTRS shall use a “flow control” security policy.

System High Access Control Database

- The system high access-control database shall support single level objects on the Red-side of the JTRS.
- All possible (allowable) message traffic shall be identified within the security policy in the operating environment.

MILS Access Control Database

- The MILS access control database shall include labels that include security classification and security compartmentalization.
- Each GPP shall permit applications at a single security classification level within a single compartment (if any compartments exist).
- Single classification level enforcement on each GPP shall be provided by the security policy within the operating environment.
- The security policy shall permit only single level objects within the system. Objects include software, hardware, and input/output devices (e.g., Ethernet port).

5.4.3 Intrusion Tools

Description: Intrusion detection allows the system to identify when an unauthorized user has gained access to the system, or an authorized user has gained access to a portion of the system that was not within his privileges. Intrusion detection is a sub-class of access tools.

Architecture/Implementation: Implementation.

Place in the Architecture: Not applicable.

Implied Requirements:

- Rules establishment and Audit

5.4.4 Virus Detection Tools

Description: Virus detection allows the system to identify when software executables have been altered.

Architecture/Implementation: Implementation.

Place in the Architecture: Not applicable.

Implied Requirements:

- Verify software integrity (e.g. CRL, secure Hash)

5.5 SYSTEM APPLICATION FUNCTIONS

5.5.1 Declassified Box

Description: A declassified box is a terminal that has been placed in the Cryptographically Controlled position.

Architecture/Implementation:

Place in the Architecture:

Implied Requirements:

- Key splits
- Generate Random
- Erase Red memory
- Zeroize Red memory
- Erase Classified application

5.5.2 System Startup

Description: The JTRS performs self-test on all processors within the system. The processors report test status to the radio self-test application. Cooperative tests between processors can then be initiated. Each processor reports the cooperative tests results to the radio self-test application. The JTR radio becomes ready for application loading after it is determined sufficient radio assets exist.

Architecture/Implementation: Each architectural element shall perform its startup test sequence.

Place in the Architecture: Core framework or application

Implied Requirements:

Normal (Token in, Switch ON)

- Initialize (BOOT) System Test
- Update Token
- Pr-Load Configuration file (punch in or call file waveform)
  - Match I/O ports to waveform
  - Load Parameters
- Idle channel

- Run
- Monitor
- Store tables and applications
- Idle
- Shut down (normal return to off)

Part Of shut down process (Normal, During normal shut down box gets declassified)

- Store Tables and applications
- Erase RED memories and applications
- Overwrite RED algorithms
- Zeroize RED key
- Zeroize all TRANSEC keys
- Remove switch key

If Token is removed (Switch still in ON position)

- Overwrite RED Algorithm
- Zeroize RED keys
- Erase RED memories
- Disable Decrypt
- Inhibit CT outputs
- Zeroize all TRANSEC Keys

Standby

- No TX or RX
- Power Savings (only power necessary functions)
- Declassify
- Capability to retain Key(s)
- Hot Boot (could be classified as standby)

Power down (Anomalies)

- Under voltage detection
- Transient detect
- Tamper Detect

5.5.3 Identify Resources (Status Available Algorithms/Key Configuration Management)

Description: The JTRS radio configures virtual channels based on resource availability. Each architectural element provides the resources available to the Device Manager upon startup. Unique INFOSEC capabilities includes encryption/decryption algorithms available, Tag generator and traffic, and key encryption keys available (TEKs and KEKs). The TEKS and KEKs must include a reference to the algorithm where they are used.

Architecture/Implementation: The JTRS radio shall determine the algorithm and key material available upon system startup. This information shall be provided to the Domain Manager by the INFOSEC subsystem. Changes to the algorithm and key material status shall also be reported to the Domain Manager. The INFOSEC shall support queries by the Domain Manager of the current algorithm and key data available.

Place in the Architecture: Core Framework

Implied Requirements:

- The INFOSEC shall be capable of determining current resources available at startup.
- Failure to identify resources shall cause an alarm condition to be generated.
- The CF shall generate an alarm if the resource status information is not provided by the subsystems.
- The INFOSEC function shall be capable of identifying algorithm and key material when data is located in bulk storage.
- Initialize
- Configure

5.5.4 Pick Algorithm – Mode of Algorithm

Description: Several of the algorithms used within the JTRS have multiple modes. The performance requirements for waveform operation require that the algorithm mode be switched without having to reload an algorithm to achieve a different mode of that algorithm.

Architecture/Implementation: Programmable encryption devices, used within the JTRS architecture shall be configured with all modes of a specific algorithm on each virtual channel that is instantiated. This implementation shall allow switching of algorithm modes.

Place in the Architecture: N/A

Implied Requirements:

- Modes of the algorithm can be switched on a message to message basis.

5.5.5 Frequency HOPSETs and LOCKOUT Sets

Description: Applications on the black side of the JTRS architecture require the use of frequency information to tune the RF. Frequency HOPSETs specify collections of frequencies to be used by a waveform net. Lockouts specify portions of the band that are not to be used by the waveform net. A lockout set is directly correlated with a frequency hopset.

When a virtual circuit is created, the circuit will have associated with it a hopset and optionally a lockout set. This information is traditionally loaded into a radio using the Fill Port Interface. The JTRS architecture shall continue to support this implementation, directing the frequency fill information from the INFOSEC subsystem to the Black Subsystem.

Architecture/Implementation: The INFOSEC subsystem shall accept the input of frequency data to the fill port. The INFOSEC subsystem shall provide the frequency fill information to the Black Side Link Application. Distribution and control of the frequency information shall be performed on the Black side of the radio.

Place in the Architecture: Black Application, INFOSEC subsystem, File System

The JTRS architecture provides the interface for passing the hopset and lockout data outside of the INFOSEC boundary.

Implied Requirements:

- DS-101/102 fill interfaces shall be used to load fill information.
- The INFOSEC subsystem shall manage the identification and distribution of fill information to the red and black applications.
- The radio operator is required to place the radio into the fill mode prior to initiation of the fill process.
- The radio operator shall provide the label used to associate the frequency information with a net.
- The radio operator shall indicate whether the fill information is intended for immediate use or is intended for bulk storage destination.
- The radio shall maintain frequency parameters for a duration consistent with the storage requirements of cryptovariables (72 – 144 hours).
- The architecture shall support the loading of HOPSETs and LOCKOUT Sets via a user interface.
- User Interface
- Handle classified Data

5.5.6 Establish Waveform Parameters

Description: A set of parameters is required to complete a waveform instantiation. (These parameters includes a net identifier, keys, lockouts etc.) An operator may select these parameters at the operator GUI or select a configuration file that contains the required information.

Architecture/Implementation: After the resources have successfully been allocated to a waveform, parameters are passed to the application. These parameters are entered by the operator on a per channel basis.

The architecture shall support the parameter selection required for waveform instantiation. The architecture shall support the operator or file parameter configuration.

Place in the Architecture: Domain Manager, Application profiles

Implied Requirements:

- Waveform parameters are established after the Domain Profile includes successful association of the resources to the waveform. These resources encompass red, black, and INFOSEC architectural elements.
- Creation and establishment for configuration files, Tables, …

5.5.7 Establish Data Path

Description: The JTRS radio establishes circuits by connecting resources. These circuits establish the data path for a given channel. (E.g. DAMA with a Teletype user I/O)

Architecture/Implementation:

Place in the Architecture: Domain Manager

Implied Requirements:

- Domain Manager shall establish the circuit.
- Build/Establish the file for execution by the Core Framework

5.5.8 Security Monitor

Description: The Security Monitor function provides added assurance to the proper operation of hardware relative to the execution of software.

The following functions are examples of monitor logic:

Software Flow Monitor – monitor execution of proper instruction sequences for instructions involved in critical operations such as data movement to and from the encryption device and during subsystem health checks.

Data Flow Monitor – monitors specific address ranges being accessed by the processor and by locking out memory operations under the detected failure conditions.

Software Flow and Data Flow Alarm Check Logic – generates simulated error cases for Software and Data Flow monitors to detect as part of the Alarm/Alarm Checking process.

Alarm Restriction/Notification Logic - restricts operations and notifies alarm conditions to the Red and Black interfaces

Architecture/Implementation: The Security Monitor Function shall be within the INFOSEC boundary and RED subsystem. Architecturally, the extension of this concept, to provide Software Flow Monitoring and Data Flow Monitoring in support of MLS, would require the Monitor Function affected red subsystems as a minimum.

Place in the Architecture: INFOSEC boundary, Red Subsystems

Implied Requirements:

- Monitoring of software functions is required to provide redundancy.
- The security monitor shall support the enforcement of separation and execution

5.5.9 Function: Status

Description: All subsystems generate status information. The JTRS radio will provide a function for collection of status in order to determine health of resources. Status reports indicating resource failure are used to restrict operation of the radio. Status is reported at power up and throughout the power-on state of the radio.

Architecture/Implementation: The JTR architecture shall support the collection of status.

Place in the Architecture: Domain Manager/Application

Implied Requirements:

- Several elements of the Core Framework shall work in conjunction with each other to remove failed channels.
- Faults detected shall result in restriction of radio operations.
- Status shall be monitored at all times. Corrective action shall be initiated when a failure is reported.
- Each subsystem shall monitor its own status.

  The monitor function shall report status to a centralized status monitor.
- The monitor function shall report all Alarms (e.g. PTD power, TAMPER, alarm…)

The status function shall classify failures and associate actions with failures in accordance with loaded security policy. The subsystem status function shall initiate restriction of local operations when the failure classification indicates this is the proper response. The centralized status function shall initiate status response when a failure is reported. Failures reported shall be logged.

5.5.10 Built-In Test

Description: Built-In Test (BIT) is performed at power-on in order to allow the subsystems to develop a healthy resource list. In addition to containing subsystem tests, there is a need for cross-subsystem tests, to insure that the elements required to instantiate a thread are available. Cross-subsystem tests will be controlled from a central location.

Background tests will be performed continuously, with status generated when an error is detected. Built-In Test can be initiated by the operator. The operator will have the capability to select the level of testing to be performed.

Architecture/Implementation: Implementation

Place in the Architecture: All subsystems contained within the JTRS radio. Domain Manager (application)

Implied Requirements:

- Failure of resources shall result in restriction of radio operations.
- Operator commanded Built–In Test shall require the operator to place the radio into a test mode (no traffic).
- Background BIT tests shall be non-intrusive to normal operation of the radio.

- Instantiated channel test

BIT shall be performed when power is applied to a subsystem. BIT status shall be reported to a central BIT control function for the radio. The central BIT function shall initiate cross-subsystem tests once the individual subsystem tests have been performed. BIT results shall be used to populate the resource tables at power-on. Subsequent detection of faults shall update the resource tables. BIT results shall be logged by the radio.

5.5.11 Human Computer Interface

Description: The HCI function provides access control to the radio. Because different users have different roles with respect to the radio, the access control mechanism includes a segmentation of operations allowed.

The HCI provides mode control to the radio. The HCI provides the operator the ability to select and instantiate a waveform.

Visual alarms are sent to the HCI for display on the user console. Audio alarms are also sent to the HCI via an audio interface.

Architecture/Implementation The architecture must support the messaging between the HCI and the system. HCI is located on the RED/BLACK side of the radio. The HCI supports the operator for radio configuration and human data entry, which includes biometric elements.

Place in the Architecture: The HCI mechanism shall be available to a local user or a remote user. Domain Manager – Core Services Access Control *(new)*, CORBA interface – Message.

Implied Requirements:

- Use of access control mechanism based on Token implies a database of valid users must be maintained and distributed.
- Maintenance and distribution of a Token privilege.
- The architecture is the authentication of a user, that list can be stored in a protected fashion, and that the list can be updated to add and remove users by a system administrator.
- Subsystems shall report alarm conditions that result in display or audible alarm to the HCI
- The radio will lock out a user who is not authenticated.
- The lockout method must be able to be cleared by a user with appropriate privileges.
- Authentication of remote user/commands

5.5.12 Encrypt/Decrypt storage

Description: The INFOSEC needs to provide bulk/selective en/decryption services for user apps or CF.

Architecture/Implementation: Impacts INFOSEC Messaging with the CF. CF needs to allow this type of messaging.

Place in the Architecture: The en/decrypt services of the INFOSEC shall be capable of bulk en/decryption as required.

Implied Requirements:

- Look at implied requirements on Encrypt/Decrypt.

5.5.13 Storage Commands – Human Control Interface

Description: The SCAS needs to support messaging so that the operator can perform the following functions:

Load fill, Load algorithms, select fill, select algorithm, view INFOSEC resource list of fill and algorithms, etc.

Architecture/Implementation: The CF shall support the messaging required for the INFOSEC /HCI functionality.

Place in the Architecture: Radio Control

Implied Requirements: None

5.5.14 Storage Control – Verify (INFOSEC Configuration Control)

Description: The JTR must have a methodology for assuring security critical control has been accomplished with the proper notification to the user. A continuos check of this control is also required.

Architecture/Implementation: Need for a control validation object shall exist while the JTR is running. Storage control validation is a security function that shall constantly check that the current control matches the requested control.

Place in the Architecture: It needs to reside with the INFOSEC or with the Domain Manager.

Implied Requirements: None

5.5.15 Storage Control – Process (INFOSEC Configuration)

Description: The JTR must have a methodology for assuring security critical control has been accomplished. Security critical control includes selection of algorithm and keys. It also includes selection of a plain text bypass function and the command bypass features that need to be validated.

Architecture/Implementation: A control validation object needs shall exist when the JTR is running. It is a security function that shall constantly check that the current control request is allowable by the operator.

Place in the Architecture: It needs to reside with the INFOSEC or with the Domain Manager.

Implied Requirements: None

5.6 SYSTEM FUNCTIONS

5.6.1 Establish/Maintain Path (Security Related)

Description: The JTRS radio is a multi-channel capable. Each channel needs to operate independently. Each channel needs to be established as a virtual connection between User I/O, Red Applications, INFOSEC, Black Applications, Modem and RF.

Architecture/Implementation: The JTRS radio shall be capable of instantiating a channel that establishes a virtual path between the User I/O, Red Applications, INFOSEC, Black Applications, Modem and RF.

Place in the Architecture: Core Framework/Application?

Implied Requirements:

- The establishment of virtual channels shall be independent of the Hardware Architecture.

5.6.2 Memory

The MSRC wants to keep security functions out of the Core Framework. The MSRC also wants to use COTS S/W were applicable to take advantage of technological improvements and promote commercial acceptance of the MSRC architecture. It is desirable to have all memory functions that are security critical, be executable applications.

5.6.2.1 Memory Allocation (Security Related)

Description: Before a waveform is instantiated, the radio will check for sufficient memory at all processing nodes. Done in the Domain Manager

Architecture/Implementation: Prior to waveform instantiations, the Device Manager needs the target hardware memory profile. The Memory Management Unit, (MMU), maintains the memory map, which delineates used and unused memory.

Place in the Architecture: Operating System (OE)/Application

Implied Requirements:

- The destination H/W needs to identify various assets that are available.
- The amount of usable memory is one of these assets.

5.6.2.2 Memory Separation

Description: There are two kinds of memory separation. Memory can separate executable objects or traffic data.

Architecture/Implementation: Architecture must guarantee ownership of memory space for executable objects and/or tasks that exits in a memory space. A rule set will define constraints. The architecture must notify the operator of any rule violation.

Place in the Architecture: All red side processing nodes must be capable of separating executable objects and traffic data.

Implied Requirements:

- If several tasks are executing, is it a requirement to separate these tasks if the tasks are in themselves not security critical although the tasks may be manipulating classified data traffic? Is it good enough to guarantee separation of classified data?

5.6.3 Memory Erasure (Object Reuse)

Description: The reallocation or tear down of an instantiated algorithm. It also refers to the elimination of red traffic data from red applications.

Architecture/Implementation: If a cryptographic resource is no longer required, the INFOSEC shall purge this function from memory.

Place in the Architecture: INFOSEC and Red traffic data processing on all red processing nodes.

Implied Requirements:

- When a processing node has finished with the red traffic data, the memory needs to be overwritten.

5.6.4 Power

Primary power shall be filtered in accordance with TBD to prevent EMC/EMI radiation.

5.6.4.1 Power Red/Black

Description: The Radio shall have a well-defined Red/Black boundary.

Architecture/Implementation: The Power Subsystem, (S/S), shall be able to isolate logical Red noise from the Black power.

Place in the Architecture: The requirement primarily resides in the Power S/S. (May change if local power regulation is used.)

Implied Requirements: None

5.6.4.2 (Power) Voltage

Description: Independent voltage detection scheme to protect the cryptographic logic from operating during an over voltage, (OV), or under voltage, (UV) event.

Architecture/Implementation: OV/UV protection shall be provided on all voltages for security critical logic. Upon detection of an OV/UV event, the SCAS shall support a quick shutdown of all cryptographic critical functions.

Place in the Architecture: INFOSEC

Implied Requirements:

- An OV/UV event must have priority over pending tasks. Nothing in the architecture can stop the shutdown.
- Notification to the HCI.

5.6.4.3 Protection (Power)

Description: The power subsystem needs to protect the radio from power surges, spikes and dropouts.

Architecture/Implementation: The Power subsystem shall protect the radio from damage in the event of power surges, spikes and dropouts in accordance with TBD.

Place in the Architecture: Power subsystem

Implied Requirements:

- Notification to the HCI.

5.6.5 Parameter Storage

Description: Storage of any system black data.

Architecture/Implementation: The CF shall support a file system for storing and retrieving information.

Place in the Architecture: Core Framework

Implied Requirements:

- Any subsystem, including INFOSEC needs the ability to store black information in bulk storage.

5.6.6 Interface Hardware/Software (external)

Description:

Architecture/Implementation:

Place in the Architecture:

Implied Requirements:

5.6.7 Reliability

Description:

Architecture/Implementation:

Place in the Architecture:

Implied Requirements:

5.6.8 Availability

Description:

Architecture/Implementation:

Place in the Architecture:

Implied Requirements:

5.6.9 TEMPEST

Description: Undesirable Radio emanations

Architecture/Implementation: The JTR shall meet TBD.

Place in the Architecture: Distributed

Implied Requirements:

5.6.10 TAMPER

Description: Unauthorized box entry.

Architecture/Implementation: The JTR radio shall include the detection and reaction to unauthorized intrusions.

Place in the Architecture: Distributed

Implied Requirements:

5.6.11 Physical Security

Description:

Architecture/Implementation: The JTR shall employ best design practices to prevent unauthorized/undetected access to the internal radio components.

Place in the Architecture: Mechanical

Implied Requirements:

6 COMMON SERVICES AND DEPLOYMENT CONSIDERATIONS

6.1 COMMON SYSTEM SERVICES

There has been no common services identified at this time. Therefore, no rationale has been generated.

6.2 OPERATIONAL AND DEPLOYMENT CONSIDERATIONS

There has been no common interfaces or features identified at this time. Therefore, no rationale has been generated.

7 ARCHITECTURE COMPLIANCE

This section provides linkage between operational requirements of the JTRS ORD and architecture requirements in the SCA. These are very different types of requirements documents: the ORD is motivated by the needs of operational forces and identifies capabilities required for future communications. The SCA constrains the design of systems that are intended to meet these operational requirements by reducing the cost of applications software portability and readily permit upgrades to new waveforms to meet future operational needs.

The tables list ORD requirements (Basic ORD and Domain Annexes) and JTRS Program Objectives, show their category (KPP, Threshold, Objective, or Not Labeled), list those that were identified as possible architecture drivers in Step 1 of JTRS (requirements that engineering experience indicated might be difficult to implement in a JTRS Architecture constrained design), identify "architecture influencers" assess the influence (high, medium or low); and point to the SCA section or paragraph that pertains to the requirement. (see definitions below).

The final column provides rationale for the linkage from two viewpoints:

how the requirement caused or directly relates to the SCA element, and

given the SCA requirement statement, how does it meet (and/or allow to be met) the ORD requirement.

This column is filled for all requirements identified as influencing the architecture.

The following definitions were used:

Architectural Influencers – Requirements from the JTRS ORD or JTRS Program objectives from the Step 1 BAA that caused features of the architecture to be included. For example the program objective of "S/W independence from H/W" influenced the architecture to include the CF, CORBA and POSIX to isolate the software from the hardware processor.

Performance Difficulty – an engineering assessment, based primarily on experience with legacy implementations, of the difficulty that an implementer might have in meeting an ORD requirement using the JTRS architecture. Assessments are either "Low", "Medium", or "High".

Table 7-1: ORD Architecture Drivers

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(1)(a)	The JTR architecture shall be capable of supporting secure and non-secure voice, video and data communications using multiple narrow-band and wideband waveforms as specified in paragraph 7, tables 1 and 2, identifying threshold KPP, threshold, and objective requirements for FY`00 through FY`05, including future waveforms as they are developed.	KPP	Yes	High	Sec 2.2.2.2 (footnote); 2.2.1.5.7; 3.1.3.2.2.5.1.3; Table 3-1; Sec 5 (V 2.0) and Security Supplement	This requirement added security examples in section 2 and 3. Sec 5 of SCA V2.0 will contain requirements that are identified as required for the operating environment. The Security Supplement will contain APIs for the Security entity. These elements will standardardize the JTRS approach for security implementation and be used in the secure modes of the ORD waveforms.
4.a.(1)(b)	The JTR program shall provide an internal growth capability through an open systems architecture approach.	KPP	Yes	Low	Sec 4.5.1, 4.5.2, 4.5.2.2, JTRS API Supplement	The SCA is based on many open system standards for it requirements. Even in those instances where a vender selects from a particular standard hardware interface and includes unique connections between hardware modules, the SCA requires those to be published when used so that others may reuse the hardware. The API definitions will publish standard interfaces available for application software. Each compliant software application publishes the interfaces to the operating environment and makes them available for reuse for other applications and upgrades.
4.a.(1)(b)	The growth capability shall be in compliance with the Joint Technical Architecture.	KPP	Yes	Low	Sec 1.2.1	The SCAS is compliant with the JTA.
4.a.(1)(b)	The growth capability shall be modular, scaleable, and flexible in form factor (threshold) (KPP).	KPP	Yes	Med	Sec 3.1, 3.2, 4.5	4.5 General hardware rules do not restrict form factor, either at the module level or at the chassis level, to permit flexible implementation to fit the specific needs of procuring agency. 3.1 & 3.2 The software architecture (Operating system, CORBA, Core Framework, and application requirements) collectively provides a modular and scalable environment for software applications. The use of COTS, both hardware and software, aids in the domain's ability to be both modular and scalable.

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(1)(c)	The JTR shall provide the operator with the ability to load and/or reconfigure modes/capabilities (via software) while in the operational environment (threshold)(KPP).	KPP	Yes	High	3.1.3.4, 3.2.2.1. B.3.1.1, B.3.1.2, D	Radio reconfiguration in the operational environment utilizes application interfaces such as the installer utility with its attendant CF DomainManager, DeviceManager, and File Manager with specific POSIX behavioral single-process and multi-processes. The concept and implementation of the Domain Profile are direct results of this requirement.
4.a.(1)(d)	The JTR shall have the ability to be reconfigured (hardware changes/upgrades) in the operational environment (threshold).	Threshold	Yes	Low	4.5.1, 4.5.2, 4.5.4, 3.2.2.1.1, D3	4.5.2 Published definition of hardware critical interfaces allows hardware modules to be replaced in an operational environment with other hardware modules meeting those published interfaces. 4.5.1 Device profiles provide hardware characteristics for registration with the core framework. 4.5.4 Hardware modularity facilitates replacement in an operational environment. Hardware configurations of a domain are maintained by their Hardware profile, Device Package Descriptor D3. These profiles are administered by the installation utility of the SCA.
4.a.(1)(e)	The JTR shall be capable of operating in a radio frequency spectrum from 2 MHz to 2 GHz suitable for the particular set of waveforms that the JTR will support (threshold) (KPP).	KPP	Yes	No		
4.a.(1)(e)	The JTR shall be capable of incorporating military and commercial satellite and terrestrial communications above 2 GHz (objective).	Objective		No		
4.a.(1)(f)	The JTR shall have the ability to retransmit/cross-band information between frequency bands/waveforms supported (threshold)(KPP) Maritime/Fixed Station Domain (Objective).	KPP	Yes	Med		The SCA does not define performance for specific waveforms or capabilities.

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(1)(g)	The JTR shall be capable of operating on multiple full and/or half-duplex channels at the same time (threshold)(KPP).	KPP	Yes	High	3.2, B.3.1.2	RF channels, whether half or full duplex, are instantiated by waveform applications as defined in SCA section 3.2. The number and availability of channels is governed by the ORD-specified domain. Simultaneous operation of multiple channels becomes a function of the performance of the domain hardware. A multi-process OS is defined by the SCA and is detailed in appendix B.3
4.a.(1)(h)	The JTR shall have the capability of automatic protocol conversion and message format translation of voice, video, or data between frequency bands or waveforms as specified in paragraph 7, Table 1 (threshold).	Threshold	Yes	No		
4.a.(1)(i)	Without interfering or overriding terminal operations, the JTR shall be capable of distributing and accepting software upgrades that have integrity and can be authenticated when transmitted through the network with which it interfaces (threshold).	Threshold	N	High	Sec 3.2.2.1; 4.2.3.5	The SCA provides for both the Installer Utility and the physical interface that would use it. The Installer and interface are specific to the target domain and platform. The SCA does not define performance required to meet the non-interference requirement. That depends on implementation specific hardware and software.
4.a.(1)(j) 1	comply with applicable National, and International spectrum management policies and regulations(threshold).	Threshold	N	No		
4.a.(1)(j) 2	be mutually compatible with other electric or electronic equipment within their expected operational environment (threshold).	Threshold	Yes	No		
4.a.(1)(k)	The JTR shall provide the ability to scan a minimum of 10 operator designated fixed frequencies or presets (threshold).	Threshold	N	No		

ORD						
ORD Paragraph Number	OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
4.a.(1)(l)	The JTR or its installation kit shall provide domain specific interfaces to ancillary equipment in order to minimize platform integration impact, i.e., power amplifiers, power supplies, antenna couplers, antennas, etc. as stated in the annexes (threshold).	Threshold	N	No		
4.a.(1)(m)	The JTR shall employ protective measures against electromagnetic pulse (EMP) and directed energy threats (objective).	Objective	N	No		
4.a.(1)(n)	After an unexpected power loss, and upon restoral of power, the JTR shall be capable of completing a components diagnostics test and a systems recovery to include: hardware, software, presets and settings (threshold).	Threshold	N	Low	Sec 3.1.3.1.3	The SCA provides for testable objects that can perform any required diagnostics tests. The SCA also provides for but does not describe an HCI Client that interfaces to the Domain Manager. The HCI can be implemented to retain knowledge of the operating state that enables it to direct the CF to configure to any known state. Knowledge of the nature of a power loss - expected vs. unexpected - is a HCI Client implementation decision and is not covered in the SCA.
4.a.(1)(o)	The JTR shall provide a standard interface to exchange voice/video/data with Service host systems (threshold).	Threshold	N	Low	Sec 1.2.1; JTRS API Supplement	The JTA imposes standard interfaces for these data exchange types. No specific requirements on the external interfaces of the waveform application are included in the SCA. However, the interface standards imposed by the JTA apply to JTRS as a C4I system.
4.a.(2)(a)	The JTR shall provide a scaleable, embedded programmable cryptographic capability, which will support black key (threshold).	Threshold	N	Med	Sec 2.2.1.7.5; 4.2.3.4; JTRS Security Supplement	The SCA does not define how INFOSEC requirements are to be implemented. It does isolate cryptographic functions so that programmable implementations can be used inside this entity. The JTRS Security Supplement incorporates the requirements for implementing black key fill.
4.a.(2)(b)	The JTR shall provide transmission security (TRANSEC) capability (threshold).	Threshold	N	No		

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(2)(c)	The JTR shall be capable of interfacing with an National Security Agency (NSA) approved electronic key management system (EKMS) (threshold).	Threshold	Yes	Low	Sec 4.2.3.4	The SCA does not define performance for specific waveforms or capabilities. Specifically, EKMS extends well beyond the scope for any specific JTR product. The SCA does define a hardware INFOSEC Class that has a key fill type attribute that permits implementation of hardware to interface in an EKMS compliant manner when such compliance is defined.
4.a.(2)(d)	The JTR shall be capable of using over the air rekeying (OTAR)/zeroizing (OTAZ)/ transfer (OTAT) as implemented by the Key Management Authority (threshold).	Threshold	Yes	Med	Security Supplement	This capability will be defined in the JTRS security supplement. Implementation of this capability has NSA participation and will conform to NSA guidelines.
4.a.(2)(e)	The JTR shall be capable of remote identification and exclusion (lockout) (threshold).	Threshold	N	Low	3.2.1, 3.2.2.1.1	Remote identification and exclusion are a function of a waveform's ECCM capability. A JTRS channel is intsantiated as that type of waveform by installing it's application and executing it.
4.a.(2)(f)	The JTR shall be capable of supporting encrypted Global Positioning System (GPS) (threshold).	Threshold	N	Low	Sec 2.2.1.7.6 Figure 4-2	The SCA provides for a GPS Hardware Child Class. The UtilityResource can provide an implementation-defined SitAwareResource Type that provides GPS-derived information to the waveforms. Future versions of the SCA may define the API for the UtilityResource and the SitAwareResource
4.a.(2)(g)	After a primary power loss, the JTR shall be capable of retaining perishable cryptographic variables for at least 72 hours (threshold)	Threshold	N	No		
4.a.(2)(g)	After a primary power loss, the JTR shall be capable of retaining perishable cryptographic variables for at least 144 hours (objective).	Objective	N	No		
4.a.(2)(h)	The JTR shall be capable of implementing public cryptography to provide privacy (objective).	Objective	No	Low	Sec 2.2.1.7.5; 4.2.3.4; JTRS Security Supplement	Cryptographic functions including algorithms used and key fill method are isolated in the SCA to the INFOSEC entity within the architecture. Specific implementations of the type of key and cryptographic method used are not part of the SCA, but are permitted in these implementations. Security policy and requirements are specified in the JTRS Security Supplement to v 2.0

ORD						
ORD Paragraph Number	OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
4.a.(2)(i)	The JTR shall provide for INFOSEC and protection of data at multiple levels of Security from Unclassified through Secret (objective).	Objective	No	High	Security Supplement	MLS capability is defined in the JTRS security supplement. Implementation of this capability has NSA participation and will conform to NSA guidelines.
4.a.(2)(i)	The JTR shall provide for INFOSEC and protection of data in a Secret High network (threshold).	Threshold	Yes	Med	Security Supplement	This capability will be defined in the JTRS security supplement. Implementation of this capability has NSA participation and will conform to NSA guidelines.
4.a.(2)(j)	In conjunction with waveforms for JTR domain family members in Paragraph 7, Tables 1 and 2, the JTR shall be capable of supporting cryptographic functions which operate with or have an external interface to cryptographic systems as listed in Annex D (threshold).	Threshold	Yes	Med	Sec 4.2.3.4	The SCA does not define performance for specific waveforms or capabilities. The SCA does define a hardware INFOSEC Class that has attributes sufficient for all required cryptographic functions.
4.a.(2)(k)	The JTR shall be capable of being handled as an unclassified Controlled Cryptographic Item (CCI) when the embedded device that provides security is not keyed (threshold).	Threshold	N	No		
4.a.(2)(l)	The JTR shall have the capability to zero locally by requiring at least two discrete operator actions to reduce the probability of accidental zeroing (threshold).	Threshold	N	No		
4.a.(2)(m)	The JTR shall employ the Defense Information Infrastructure (DII) Key Management Infrastructure in supporting JTR integrity, identification, and authentication requirements (threshold).	Threshold	N	Low	JTRS Security Supplement	The JTRS Security Supplement (to be updated in V2.0) imposes requirements derived from the DII Key Management Infrastructure. Material contained in the SCA on key management has been coordinated with the NSA.

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(2)(n)	Each JTR shall be equipped with the capability to receive Global Positioning Satellite (GPS) signals. This GPS receive capability shall be in addition to the number of channels for each JTR category specified by the Domain Annexes.		N	Low	Sec 4.2.2, 4.5	The SCAS supports installation of GPS receive capability.
4.a.(3)	The JTR shall be capable of providing scaleable networking services for connected RF (over the air) networks, host networks, and hybrid networks in Increment 3 and in accordance with the phased procurement implementation specified in paragraph 7 (threshold) (KPP).	KPP	N	Med	Sec 2.2.2; JTRS API Supplement	Networking requirements are met by the operating environment and application software running in that environment and are not specified within the SCA. The SCA does standardize APIs for network applications and can host waveforms that have scalable networking capability (e.g., VRC-99)
4.a.(3)(a)	The networked JTR shall extend, between and across the geographical and/or organizational boundaries within a nominal area of operations (threshold) (KPP).	KPP	N	No		
4.a.(3)(b)	The networked JTR shall provide a scaleable and interoperable means to establish point-to-point (two way), multi-point (two way), multicast (up to 100 selected nodes), and broadcast data capability between/among any user-selected nodes in a joint network (threshold).	Threshold	N	Med	Sec 2.2.2; API Supplement	Networking requirements are met by the operating environment and application software running in that environment and are not specified within the SCA. The SCA does standardize APIs for network applications and can host waveforms that have scalable networking capability (e.g., VRC-99)
4.a.(3)(c)	The networked JTR shall provide for mobile JTRs to readily transfer between authorized networks. This transfer should be transparent to the user. (threshold).	Threshold	N	Low	3.2.1, 3.2.2.2	3.2.1 The SCA supports the hosting of various networking protocols as applications. 3.2.2.2 Seamless transfer of data is supported through the use of the Service API's. Service API's include Network, Security and I/O services for the domain.
4.a.(3)(d)	The networked JTR shall provide routing capability, interface connectivity that extends into the Internet Protocol, military packet networks (threshold).	Threshold	N	No		
4.a.(3)(d)	The networked JTR shall provide routing capability, interface connectivity that extends into the Internet Protocol and/or cell networks (objective).	Objective	N	No		

ORD						
ORD Paragraph Number	OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
4.a.(3)(e)	The networked JTR shall perform dynamic intra-network and inter-network routing for data transport based on priority (threshold).	Threshold	N	No		
4.a.(3)(f)	The networked JTR shall include hardware and software sufficient to organize, manage, and dynamically control network connectivity structures, routing mechanisms, and bandwidth allocations (threshold).	Threshold	N	No		
4.a.(3)(g)	The networked JTR shall selectively transmit individual location information to selected JTR nodes.	Threshold	N	No		
4.a.(3)(g)	The networked JTR transmission shall be passed in the Military Grid Reference System and/or in the latitude and longitude to a host system (threshold).	Threshold	N	No		
4.a.(3)(h)	The JTR system shall provide network management capability to respond to changes in mission or organization, and reconfigure at a minimum a battalion-sized joint task force network (with approximately 150 JTRS terminals), in 15 minutes (threshold, KPP).	KPP	N	Low	Sec 1.2.1; JTRS API Supplement	JTRS implementations are C4I systems that will comply with JTA data exchange and network management protocols. These protocols provide for network link status. Applications waveforms that provide wireless networking capability will be required to support network management requirements. These requirements are not part of the SCA, but would be application waveform performance requirements.
4.a.(3)(h)	The JTR system shall provide network management capability to respond to changes in mission or organization, and reconfigure at a minimum a battalion-sized joint task force network (with approximately 150 JTRS terminals), in 5 minutes (objective).	Objective	N	Low	3.1.2, 3.1.3.4, D	CORBA software provides rapid system reconfiguration. The Domain Profile implementation provides reconfigurability capability.
4.a.(3)(i)	The JTR network shall have the capacity to meet the information flow of waveforms/capabilities as specified in paragraph 7, Table 1 (threshold).	Threshold	Yes	High	Sec 3.2.2.2.2.2	The SCA does not define performance required to meet waveform/capabilities. That depends on implementation hardware. Stressing waveforms are identified in this SRD section and rationale provided that shows these can be implemented using the SCA with today’s technology. The SCA permits alternate transfer paths if middleware/overhead limits performance.

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(3)(j)	The JTR shall provide information to Service and joint network management tools to assess and report network link status (threshold).	Threshold	N	Low	Sec 1.2.1; JTRS API Supplement	JTRS implementations are C4I systems that will comply with JTA data exchange and network management protocols. These protocols provide for network link status. Applications waveforms that provide wireless networking capability will be required to support network management requirements. These requirements are not part of the SCA, but would be application waveform performance requirements.
4.a.(3)(k)	The JTR network shall provide a name-to-address translation service that supports automatic registration and de-registration of host names and addresses (threshold).	Threshold	N	No		
4.a.(3)(l)	The JTR network shall provide the capability for users to address data to other users by using position/ organization names in the address fields (e.g., S3.2AR.BDE) (threshold).	Threshold	N	Low	Sec 1.2.1; JTRS API Supplement	JTRS implementations are C4I systems that will comply with JTA data exchange and network management protocols. These protocols provide for network link status. Applications waveforms that provide wireless networking capability will be required to support network management requirements. These requirements are not part of the SCA, but would be application waveform performance requirements.
4.a.(3)(m)	The JTR system shall provide the means to support message delivery based on geographic areas (objective).	Objective	N	Low	3.2.1, 3.2.2.2	3.2.1 The SCA supports the hosting of various networking protocols as applications. Transfer of data over geographical areas is a function of the waveform application. 3.2.2.2 Application's are supported with Service API's which provide I/O and networking services.
4.a.(3)(n)	The JTR network shall provide information and be interoperable with the joint network management tool, to allow network managers to remotely identify and configure user access and profile parameters to prioritize users' network access and message delivery (threshold).	Threshold	N	Low	Sec 1.2.1; JTRS API Supplement	JTRS implementations are C4I systems that will comply with JTA data exchange and network management protocols. These protocols provide for network link status. Applications waveforms that provide wireless networking capability will be required to support network management requirements. These requirements are not part of the SCA, but would be application waveform performance requirements.
4.b	The JTR shall have the following mission-capable requirements for both wartime and peacetime.					

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.b(1)	The JTR shall have an operational availability (Ao) of 96 percent (threshold)(KPP).	KPP	N	No		
4.b(2)	The JTR hardware size and weight shall be compatible with current platforms, as specified in the domain annexes (threshold).	Threshold	N	No		
4.b(2)	The JTR hardware size and weight shall also consolidate many individual functions of current terminals into one physical chassis, thereby reducing weight and space requirements (threshold).	Threshold	N	No		
4.b(3)	The JTR shall be logistically supportable within each Service (threshold).	Threshold	N	No		
4.b(4)	The JTR internal test and diagnostic built-in-test (BIT) provisions shall be capable of fault isolation (threshold).	Threshold	N	No		
4.b.(1)	The JTR shall have an operational availability (Ao) of 99 percent (objective).	Objective	N	No		
4.c.(1)	The JTR shall provide an operator-selectable capability to operate in listening silence (receive only) mode (threshold).	Threshold	N	No		
4.c.(2)	The JTR shall effectively operate in wartime operations and in worldwide conditions as specified in the domain annexes (threshold).	Threshold	N	No		
4.c.(3)	The JTR shall be capable of being operated and maintained in a nuclear, biological and chemical (NBC) environment, as specified in the domain annexes (threshold).	Threshold	N	No		

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.c.(4)	The JTR shall be capable of providing a platform- specific human computer interface, as specified in the domain annexes (threshold).	Threshold	N	Low	Sec 2.2.1.7.6; 3.1.3	The SCA isolates user and platform interfaces through modularity of the software and hardware associated with those interfaces. Platform unique interfaces are accommodated within the User I/O hardware class and through extensions of the CF Resource class such as a UtilityResource and the Device class such as I/O Device
4.c.(5)	The JTR shall be capable of incorporating power management to achieve maximum efficiency (threshold).	Threshold	N	No		
4.c.(6)	The JTR shall be capable of withstanding power surges (threshold).	Threshold	N	No		
5	Program support for the JTR shall be in place when the initial operational capability (IOC) is achieved.		N	No		
5	Program support for the JTR shall be expanded, as necessary, for each Service prior to achieving full operational capability (FOC).		N	No		
5.a.	Operator level maintenance shall be limited to reconfiguration for needed capabilities.		N	No		
5.a.	Preventive/corrective maintenance shall be limited to the predetermined lowest repairable unit (LRU).		N	No		
5.a.	Life-cycle logistics support factors shall be implemented that provide for cost effective maintenance of the JTR.		N	No		
5.b.	Where the DoD logistics structure is used, General Purpose Electronic Test Equipment (GPETE) shall be selected from existing standard GPETE equipment lists.		N	No		
5.b.	The use of Special Purpose Electronic Test Equipment (SPETE), special purpose support equipment, and special tools shall be avoided to the maximum extent possible.		N	No		

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
5.b.	JTR BIT/built-in-test equipment (BITE) performance shall not be accomplished using GPETE, SPETE, or substituting modules.		N	No		
5.b.	For all non-developmental item (NDI) equipment the contractor shall identify automatic test equipment (ATE).		N	No		
5.b.	For contractor provided logistics support, equipment and processes shall be identified.		N	No		
5.c.(1)	JTR must be easily maintainable and operable, incorporating the principles of modularity and commonality.		Yes	Low	Sec 4.5, 3.2	The SCAS hardware rules set supports this requirement from the hardware standpoint. The required use of CORBA software supports this from the software standpoint.
5.c.(1)	The JTR shall conform to applicable human engineering design criteria.		Yes	No		
5.d.(1)	The JTR management and components shall provide checks for computer operations system viruses during systems initialization and routine operations.		N	Low		This capability will be defined in the JTRS security supplement. Implementation of this capability has NSA participation and will conform to NSA guidelines.
5.d.(1)	The operator shall be alerted to a detected virus.		N	Low		This capability will be defined in the JTRS security supplement. Implementation of this capability has NSA participation and will conform to NSA guidelines.
5.d.(2)	A software support capability shall be functional by JTR's Initial Operational Capability (IOC) and must provide for update, configuration control, and management of all computer programs and data.		N	No		
5.e.	In compliance with the Continuous Acquisition Lifecycle Support (CALS) program, JTR shall comply with specifications and standards approved within DOD for creation, use, and management of technical and other data in digital form.		N	No		

ORD						
ORD Paragraph Number	OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
5.e.	For Army and Air Force employment, logistics support should include sufficient quantities of Mobility Readiness Spares packages and Peacetime Operating Stocks for continued supportability. For the Navy and Marine Corps, spares will be based on the On Board Repair Parts requirements, as calculated for each platform. If required, spares will be pre-positioned.					
5.f.(2)	The JTR shall have adequate security safeguards and compartmentalization to ensure the confidentiality, integrity, and availability of the information passing through or residing on it. Security features of the JTR will comply with the Multi-Level Information Systems Security Initiative (MISSI).		Yes	High	Security Supplement	Implementation of this capability has NSA participation and will conform to NSA guidelines as well as comply with MISSI.
5.f.(2)	Security features of the JTR will also comply with its follow-on.					
5.f.(2)	All C4I resources will be certified for end-to-end interoperability by complying with the intent of CJCSI 6212.01A, 30 June 1995.					
5.f.(3)	This ORD has been assigned a joint potential designator (JPD) of "Joint."					
5.g.	JTR distribution and basing will be consistent with existing force structures and deployment concepts. If JTR components are integrated into other systems, transportability requirements of the host system apply.					
5.h.(1)	The JTR acquisition will adhere to the Joint Technical Architecture in identifying the standards and guidelines.					

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
5.h.(2)	Electronic keying shall be applied to allow cryptographic processes implemented by JTR. All cryptographic systems must be interoperable with the EKMS, and the envisioned Joint Key Management System (JKMS).		N	Low	Sec 4.2.3.4	The SCA does not define performance for specific waveforms or capabilities. Specifically, EKMS extends well beyond the scope for any specific JTR product. The SCA does define a hardware INFOSEC Class that has a key fill type attribute that permits implementation of hardware to interface in an EKMS compliant manner when such compliance is defined.
5.h.(2)	Only National Security Agency (NSA) endorsed and approved security products, techniques, and protective services shall be used to secure classified communications.		N	No		
5.h.(3)	The JTR must be interoperable with National Airspace System architecture.					
5.h.(4)	For operation with North Atlantic Treaty Organization (NATO) member nations, the technical characteristics of the JTR shall conform with the applicable requirements of the Standardization Agreements.		N	No		
5.h.(5)	All information technology for the JTR shall be DOD approved and where applicable, selected from those contained in the DISA approved "Profile of Standards."		N	Low	Sec 1.2.1; JTRS API Supplement	JTRS implementations are C4I systems that will comply with JTA data exchange and network management protocols. These protocols provide for network link status. Applications waveforms that provide wireless networking capability will be required to support network management requirements. These requirements are not part of the SCA, but would be application waveform performance requirements.
5.i.	When required, JTR will use National Imagery and Mapping Agency (NIMA) joint service mapping standards to ensure interoperability with other systems. Geographic mapping and gridding functions will be based on Universal Transverse Mercator (UTM).					

ORD						
ORD Paragraph Number	**OPERATIONAL REQUIREMENTS DOCUMENT (ORD) For JOINT TACTICAL RADIO (JTR)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
5.i.	Latitude/longitude coordinates referred to by the World Geodetic System (WGS-84), will be compatible with existing GPS receivers, and upgradable to future GPS receivers.					
6	Refer to the domain annexes for the Services force structure requirements. Other governmental agencies' force structure requirements will be described on a case by case basis.					
7.a.	The system will be developed incrementally providing increased capabilities as it matures.		Yes	Med	Sec 3.1, 3.2, 4.5	Generally, the features of the SCA that facilitate modularity and scalability also allows the system to be developed incrementally. These features include the use of both COTS hardware and software and the use of API's to define interface behavior of custom interfaces.
7.b.	The JTR will support the modes/capabilities depicted in Tables 1 and 2.					
7.d.	Delivery of follow-on production level articles, entailing additional objective values, are scheduled to begin in FY02 (Table 2).					
	The JTR network shall have the capacity to meet the information flow required by new capabilities and a latency near zero (objective).	Objective	No	High	Sec 3.2.2.2.2.2	Hardware limitations will impose latency on information transfer within the system. Higher speed data rates will be implemented by inserting new technology (as required) and by using alternate transfer mechanism as permitted by 3.2.2.2.2.2.
	The JTR shall provide a scaleable, embedded programmable cryptographic capability, which will support black key [for the] Maritime/Fixed Station Domain (Objective).	Objective	N	Med	Sec 4.2.3.4	The SCA does not define performance for specific waveforms or capabilities. The SCA does define a hardware INFOSEC Class that has attributes sufficient for all required cryptographic functions.
	The JTR shall provide the ability to scan individual frequency bands (objective).	Objective	N	No		

Table 7-2: ORD Annex A

ANNEX A						
ORD Annex A Paragraph Number	**ORD Annex A Airborne Domain Requirements with Table 1&2 Waveform Requirements (changes from basic ORD)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD Draft text**
4.a.(1)(a)	The airborne JTR shall meet required performance parameters when integrated into land and sea based fixed, rotary wing, and unmanned aircraft (threshold).	Threshold	N	No		
4.a.(1)(b) 1	The JTR will provide interfaces for host platform visual displays (threshold).	Threshold	N	Low	Sec 2.2.1.7.6; 3.1.3	See rationale for ORD para 4.c.(4). The SCA isolates user and platform interfaces through modularity of the software and hardware associated with those interfaces. Platform unique interfaces are accommodated within the User I/O hardware class and through extensions of the CF Resource class such as a UtilityResource and the Device class such as I/O Device
4.a.(1)(b) 2	The JTR will provide standard interfaces for host platform data input/output devices, including control and traffic platform busses (threshold).	Threshold	N	Low	Sec 2.2.1.7.6; 3.1.3, 4.2.3.5	The SCA isolates user and platform interfaces through modularity of the software and hardware associated with those interfaces. Platform unique interfaces are accommodated within the User I/O hardware class and through extensions of the CF Resource class such as a UtilityResource and the Device class such as I/O Device
4.a.(1)(b) 4	The airborne JTR will provide for remote control and operation via a remote control unit or through the host platform bus interface (threshold).	Threshold	N	Low	3.2.2.2	Control of an Airborne JTRS is expected to be via a standard bus protocol or a legacy control interface. Both of these types of interfaces are supported in the SCA by the API services defined in section 3.2.2.2.
4.a.(1)(c)	The airborne JTR shall support performance parameters while operating in the operational profile of each host airborne platform (threshold).	Threshold	N	No		
4.a.(1)(e)	In addition to GPS, the airborne JTR will provide up to six channels (threshold).	Threshold	N	Low		The SCA does not define performance required to meet waveform/capabilities. That depends on implementation hardware. The SCA does not limit the number of waveforms which can be available for deployment.

ANNEX A						
ORD Annex A Paragraph Number	**ORD Annex A Airborne Domain Requirements with Table 1&2 Waveform Requirements (changes from basic ORD)**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD Draft text**
4.a.(1)(f)	The airborne JTR shall provide the capability to choose from among at least 10 waveforms without loading additional software from an external source.(no KPP, Thres, or Obj assigned)	Threshold	N	Low	Sec 3.1.3.3.1; 3.1.3.3.2; 3.1.3.3.3	Waveform application software are contained in files stored on internal storage devices or external. The SCA CF specifies the requirements for these files. The number of applications required to be resident within a JTRS implementation will determine the capacity of the storage media for those files. The SCA does not specify capacity.
4.a.(1)(f)	The airborne JTR shall provide the capability of replacing waveforms over-the-air.(no KPP, Thres, or Obj assigned)	Threshold	N	Low	3.1.3.3.2 Security Supplement	Waveform files may be transported over the air using established channels on JTRS. At the receiving JTRS node, the files received over-the air can be managed and stored using the file system of the Core Framework in conjunction with appropriate security considerations. Once stored, the local operator can select those waveform files for instantiation.
4.a.(1)(f)	The airborne JTR shall provide the capability to choose up to 30 waveforms using a bulk storage device.(no KPP, Thres, or Obj assigned)	Threshold	N	Low	3.1.3.3.2	Waveforms in JTRS are applications as defined in section 3.2. These applications are if the form of a set of file(s). These waveform application files are supported by a federated file system, which is part of the Core Framework defined in SCA section 3.1.3.3.2. This federated file system allows the waveform files be located anywhere in the domain. This includes the mounting of an external Bulk storage device on to the domain.
4.c.(1)(a)	Integration of JTR into user platforms shall be accomplished with minimal demands for platform modifications. The JTR system shall provide radios and ancillaries that have the following characteristics:		N	Low	Sec 2.2.1.7.4 4.2.3.5	The SCA does not define interfaces required to meet specific platform requirements. That depends on implementation hardware. The SCA does define software resources and hardware classes enabling the implementation of platform-specific interfaces.
4.c.(1)(a) 1	The airborne JTR (without ancillaries) will be no larger than a ¾ long air transportable rack (ATR) (threshold).	Threshold	N	No		
4.c.(1)(a) 1	The airborne JTR (without ancillaries) will be no longer than a ½ air transportable rack (ATR) (objective).	Objective	N	No		
4.c.(1)(a) 2	The airborne JTR shall be no heavier than the radios it replaces (threshold).	Threshold	N	No		

ANNEX A

ORD Annex A Paragraph Number	ORD Annex A Airborne Domain Requirements with Table 1&2 Waveform Requirements (changes from basic ORD)	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD Draft text
4.c.(1)(a) 2	The airborne JTR shall be at least 75% lighter than the radios it replaces (objective).	Objective	N	No		
4.c.(1)(a) 3	The airborne JTR will operate off existing aircraft power systems for each platform (threshold).	Threshold	N	Low	Sec 4.2.2	The SCA isolate Power Supplies into a separate class, so that platform unique requirements can be separated from internal system power needs. This enable an airborne system acquisition authority to specify platform specific power and the JTRS system supplier to only have to change a single hardware class to accommodate the unique power of the platform.
4.c.(1)(a) 4	The airborne JTR will draw no more power than the radios it replaces (threshold).	Threshold	N	No		
4.c.(1)(a) 4	The airborne JTR will draw at least 75% less power than the radios it replaces (objective).	Objective	N	No		
4.c.(1)(a) 5	The airborne JTR shall provide interfaces to on-board automated frequency management systems (threshold).	Threshold	N	Low	Sec 2.2.1.7.6; 3.1.3, 4.2.3.5	The SCA isolates user and platform interfaces through modularity of the software and hardware associated with those interfaces. Platform unique interfaces are accommodated within the User I/O hardware class and through extensions of the CF Resource class such as a UtilityResource and the Device class such as I/O Device
4.c.(2)(a)	The JTR shall be capable of being operated and maintained in a nuclear, biological, and chemical (NBC) environment by persons in full Mission Oriented Protective Posture IV (MOPP IV) protection gear (threshold).	Threshold	N	No		
4.c.(2)(b)	The JTR man-machine interface shall be compatible with Night Vision Imaging System (NVIS) standards and shall be capable of being operated by persons wearing night vision goggles (NVG) (threshold).	Threshold	N	No		

Table 7-3: ORD Annex B

ANNEX B						
ORD Annex B Paragraph Number	**ORD Annex B_Maritime/Fixed Domain_Requirements with Table 1&2 Waveform Requirements**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
1.b.	The maritime JTR shall be part of a communications system that provides modular communicating and networking capabilities.		N	Low	3.2.1, 3.2.2.2	3.2.1 The SCA supports the hosting of various networking protocols as applications. Seamless transfer of data is supported through the use of the Service API's. Service API's include Network, Security and I/O services for the domain.
1.c.	Operational Concept (Supplemental). The maritime JTR shall be Automated Digital Network System (ADNS) interoperable.		N	Low	Sec 2.2.1.7.4 4.2.3.5	The SCA does not define interfaces required to meet specific platform requirements. That depends on implementation hardware. The SCA does define resources and hardware classes enabling the implementation of platform-specific interfaces.
1.c.	The maritime JTR shall be Advanced RF distribution systems interoperable.		N	Low	Sec 4.2.3.1	The hardware part of the SCA isolates RF external interfaces to the RF class. The Maritime domain acquisition authority would specify the interface requirements for the Advanced RF distribution systems and the system supplier would isolate those requirements to his implementation of the RF class.
1.c.	The maritime JTR shall be Submarine Antenna Distribution System (SADS) interoperable (i.e. JMCOMS interoperable).		N	Low	Sec 4.2.3.1	The hardware part of the SCA isolates RF external interfaces to the RF class. The Maritime domain acquisition authority would specify the interface requirements for the Advanced RF distribution systems and the system supplier would isolate those requirements to his implementation of the RF class.
4.a.	The JTR shall meet the following supplemental maritime specific performance parameters:					
4.a.(1)(a)	In addition to GPS, the Maritime and Fixed JTR shall provide a minimum of 4 scaleable channels (threshold).	Threshold	N	Low	4.5.1, Appendix B.3	Number and availability of channels is dependent on domain (Handheld, Maritime, etc). Simultaneous operation of multiple channels is a function of the performance of the domain hardware. A multi-process OS is defined by the SCA and is detailed in appendix B.3
4.a.(1)(a)	In addition to GPS, the Maritime and Fixed JTR shall provide a minimum of 4 scaleable channels with a growth capability to 10 scaleable channels (objective).	Objective	N	Med		The SCA does not define performance required to meet waveform/capabilities. That depends on implementation hardware.

ANNEX B

ORD Annex B Paragraph Number	ORD Annex B_Maritime/Fixed Domain_Requirements with Table 1&2 Waveform Requirements	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
4.a.(1)(b)	The JTR shall be capable of operating in sea state 5 and surviving sea state 8 on all classes of ships (threshold).	Threshold	N	No		
4.a.(1)(b)	The JTR shall be capable of operating at sea states above 5 with minimal degraded performance (objective).	Objective	N	No		
4.a.(1)(d)	The Maritime and Fixed station configuration of JTR shall provide the capability for radios to be operated, controlled, and monitored from remote locations (threshold).	Threshold	N	Low	Sec 2.2.1.7.4 4.2.3.5	The SCA does not define interfaces required to meet specific platform requirements. That depends on implementation hardware. The SCA does define software resources and hardware classes enabling the implementation of platform-specific interfaces.
4.a.(1)(e)	The JTR shall be compatible with commercial, ground mobile, and shipboard power distributed systems (threshold).	Threshold	N	No		
4.a.(1)(g)	The JTR weight shall not exceed a two person lift (threshold).	Threshold	N	No		
4.a.(1)(h)	The JTR shall have a minimum of 10 presets per channel (threshold).	Threshold	N	No		
4.a.(1)(h)	The JTR shall have a minimum of 20 presets per channel (objective).	Objective	N	No		
4.a.(1)(i)	The JTR shall provide a standard interface with legacy shipboard and fixed station communication systems (threshold).	Threshold	N	Low	Sec 2.2.1.7.6; 3.1.3	See rationale for ORD para 4.c.(4). The SCA isolates user and platform interfaces through modularity of the software and hardware associated with those interfaces. Platform unique interfaces are accommodated within the User I/O hardware class and through extensions of the CF Resource class such as a UtilityResource and the Device class such as I/O Device
4.a.(1)(j)	The Maritime/Fixed Station JTR shall provide the capability of replacing waveforms over-the-air (threshold).	Threshold	N	Low	3.1.3.3.2 Security Supplement	Waveform files may be transported over the air using established channels on JTRS. At the receiving JTRS node, the files received over-the air can be managed and stored using the file system of the Core Framework in conjunction with appropriate security considerations. Once stored, the local operator can select those waveform files for instantiation.

ANNEX B						
ORD Annex B Paragraph Number	**ORD Annex B_Maritime/Fixed Domain_Requirements with Table 1&2 Waveform Requirements**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(1)(j)	The Maritime/Fixed Station JTR shall provide the capability to choose up to 30 waveforms using a bulk storage device (threshold).	Threshold	N	Low	3.1.3.3.2	Waveforms in JTRS are applications as defined in section 3.2. These applications are if the form of a set of file(s). These waveform application files are supported by a federated file system, which is part of the Core Framework defined in SCA section 3.1.3.3.2. This federated file system allows the waveform files be located anywhere in the domain. This includes the mounting of an external Bulk storage device on to the domain.
4.a.(1)(j)	The Maritime/Fixed Station JTR shall provide the capability to choose from among at least 12 waveforms without loading additional software from an external source (threshold).	Threshold	N	No		
4.c.(1)	The JTR shall be capable of being operated in low light shipboard conditions (threshold).	Threshold	N	No		
4.c.(2)	Temperature constraints will conform with best commercial practices (threshold).	Threshold	N	No		
5.b.	JTR shall utilize the Consolidated Automated Support System (CASS).	Threshold	N	No		
5.c.	All final manpower, personnel, and training (MPT) requirements will be documented in Service training plans.	Threshold	N	No		

Table 7-4: ORD Annex C

ANNEX C						
ORD Annex C Paragraph Number	**ORD Annex C_Ground Forces Domain_Requirements with Table 1&2 Waveform Requirements**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4	All ground forces requirements for JTR, listed below, shall be based on validated rules for Operational Facilities (OPFACs) and validated Information Exchange Requirements (IERs), between OPFACs contained in the US Army Training and Doctrine Command (TRADOC) Command, Control, Communications, and Computers Requirements Definition Program (C4RDP), or Joint equivalent.		N	No		
4.a.(1)(b)	The vehicular and dismounted warfighter configurations of the JTR shall provide the capability for radios to be operated and controlled from remote locations up to 2km away (threshold).	Threshold	N	Low	Sec 2.2.1.7.4 4.2.3.5	The SCA does not define interfaces required to meet specific platform requirements. That depends on implementation hardware. The SCA does define software resources and hardware classes enabling the implementation of platform-specific interfaces.
4.a.(1)(b)	The vehicular and dismounted warfighter configurations of the JTR shall provide the capability for radios to be operated and controlled from remote locations up to 4km away (objective).	Objective	N	Low	3.2.2.2	Control of an vehicular or dismounted JTR will be via a standard bus protocol or a legacy control interface. Both of these types of interfaces are supported in the SCA by the API services defined in section 3.2.2.2.
4.a.(1)(c)	A JTR shall operate at full performance levels and not degrade mission effectiveness of host systems/platforms engaged in their operational environments, including movement and weapons firing (threshold) (KPP).	KPP	N	No		
4.a.(1)(d)	The JTR shall survive High Altitude Electromagnetic Pulse (HEMP) to the degree specified in MIL-STD-2169B but not be required to work through the event (threshold). Recycling of power to restore operation is acceptable.	Threshold	N	No		
4.a.(1)(e)	The JTR shall survive chemical, and biological attacks as well as decontamination procedures using existing solvents	Threshold	N	No		

ANNEX C

ORD Annex C Paragraph Number	ORD Annex C_Ground Forces Domain_Requirements with Table 1&2 Waveform Requirements	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
	(threshold).					
4.a.(1)(f)	The JTR shall provide a display of current own position location information at each radio (threshold).	Threshold	N	No		
4.a.(1)(g)	The JTR system shall provide operator selectable display modes that express its position in either the GPS latitude-longitude or the Military Grid Reference System (MGRS), that includes a 3-character grid zone, a 2-character 100km square, and an 8-digit map coordinate (threshold).	Threshold	N	No		
4.a.(1)(g)	The JTR system shall provide operator selectable display modes that express its position in either the GPS latitude-longitude or the Military Grid Reference System (MGRS), that includes a 3-character grid zone, a 2-character 100km square, and an 10-digit map coordinate (objective).	Objective				
4.a.(1)(h)	The JTR shall be operable and maintainable in temperatures from -40°C to +55°C (threshold).	Threshold	N	No		
4.a.(1)(i)	The JTRs shall provide the means to physically interconnect to selected external legacy radios to access the JTR network (threshold).	Threshold	N	Low	Sec 2.2.1.7.4 4.2.3.5	The SCA does not define interfaces required to meet specific platform requirements. That depends on implementation hardware. The SCA does define resources and hardware classes enabling the implementation of platform-specific interfaces.
4.a.(1)(j)	The ground forces JTR, in the Handheld configuration, shall provide the capability to choose from among at least 6 waveforms without loading additional software from an external source (threshold).	Threshold	N	High	Sec 3.1.3.3.1; 3.1.3.3.2; 3.1.3.3.3	Waveform application software are contained in files stored on internal storage devices or external. The SCA CF specifies the requirements for these files. The number of applications required to be resident within a JTRS implementation will determine the capacity of the storage media for those files. The SCA does not specify capacity. This is considered a high architecture driver for this domain because of very severe size, weight and power constraints on handheld implementations. Technology insertion of new denser and lower power electronics will reduce the impact on implementing JTRS in the handheld domain.

ANNEX C						
ORD Annex C Paragraph Number	**ORD Annex C_Ground Forces Domain_Requirements with Table 1&2 Waveform Requirements**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(1)(j)	The ground forces JTR, in the Handheld configuration, shall provide the capability of replacing waveforms over-the-air (threshold).	Threshold	N	High	3.1.3.3.2 Security Supplement	Waveform files may be transported over the air using established channels on JTRS. At the receiving JTRS node, the files received over-the air can be managed and stored using the file system of the Core Framework in conjunction with appropriate security considerations. Once stored, the local operator can select those waveform files for instantiation.
4.a.(1)(j)	The ground forces JTR, in the Handheld configuration, shall provide the capability to choose up to 30 waveforms using a bulk storage device (threshold).	Threshold	N	High	3.1.3.3.2	Waveforms in JTRS are applications as defined in section 3.2. These applications are if the form of a set of file(s). These waveform application files are supported by a federated file system, which is part of the Core Framework defined in SCA section 3.1.3.3.2. This federated file system allows the waveform files be located anywhere in the domain. This includes the mounting of an external Bulk storage device on to the domain.
4.a.(1)(j)	The ground forces JTR, in the Vehicular configuration, shall provide the capability to choose from among 10 waveforms without loading additional software from an external source (threshold).	Threshold	N	Low	Sec 3.1.3.3.1; 3.1.3.3.2; 3.1.3.3.3	Waveform application software are contained in files stored on internal storage devices or external. The SCA CF specifies the requirements for these files. The number of applications required to be resident within a JTRS implementation will determine the capacity of the storage media for those files. The SCA does not specify capacity.
4.a.(1)(j)	The ground forces JTR, in the Vehicular configuration, shall provide the capability of replacing waveforms over-the-air (threshold).	Threshold	N	Low	3.1.3.3.2 Security Supplement	Waveform files may be transported over the air using established channels on JTRS. At the receiving JTRS node, the files received over-the air can be managed and stored using the file system of the Core Framework in conjunction with appropriate security considerations. Once stored, the local operator can select those waveform files for instantiation.
4.a.(1)(j)	The ground forces JTR, in the Vehicular configuration, shall provide the capability to choose up to 30 waveforms using a bulk storage device (threshold).	Threshold	N	Low	3.1.3.3.2	Waveforms in JTRS are applications as defined in section 3.2. These applications are if the form of a set of file(s). These waveform application files are supported by a federated file system, which is part of the Core Framework defined in SCA section 3.1.3.3.2. This federated file system allows the waveform files be located anywhere in the domain. This includes the mounting of an external Bulk storage device on to the domain.

ANNEX C						
ORD Annex C Paragraph Number	**ORD Annex C_Ground Forces Domain_Requirements with Table 1&2 Waveform Requirements**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.a.(1)(j)	The ground forces JTR, in the Dismounted Warfighter configuration, shall provide the capability to choose from among 10 waveforms without loading additional software from an external source (threshold).	Threshold	N	Med	Sec 3.1.3.3.1; 3.1.3.3.2; 3.1.3.3.3	Waveform application software are contained in files stored on internal storage devices or external. The SCA CF specifies the requirements for these files. The number of applications required to be resident within a JTRS implementation will determine the capacity of the storage media for those files. The SCA does not specify capacity.
4.a.(1)(j)	The ground forces JTR, in the Dismounted Warfighter configuration, shall provide the capability of replacing waveforms over-the-air (threshold).	Threshold	N	Med	3.1.3.3.2 Security Supplement	Waveform files may be transported over the air using established channels on JTRS. At the receiving JTRS node, the files received over-the air can be managed and stored using the file system of the Core Framework in conjunction with appropriate security considerations. Once stored, the local operator can select those waveform files for instantiation.
4.a.(1)(j)	The ground forces JTR, in the Dismounted Warfighter configuration, shall provide the capability to choose up to 30 waveforms using a bulk storage device (threshold).	Threshold	N	Med	Sec 3.1.3.3.1; 3.1.3.3.2; 3.1.3.3.3	Waveform application software are contained in files stored on internal storage devices or external. The SCA CF specifies the requirements for these files. The number of applications required to be resident within a JTRS implementation will determine the capacity of the storage media for those files. The SCA does not specify capacity.
4.b.	The JTR shall be transportable worldwide (air, rail, sea, and air droppable) (threshold).	Threshold	N	No		
4.c.(1)	Integration of JTR radios and ancillaries (e.g., installation kits power amplifiers, and antennae) into user platforms shall be accomplished with minimal demands for platform modifications.	Threshold	N	Low	Sec 2.2.1.7.4 4.2.3.5	The SCA does not define interfaces required to meet specific platform requirements. That depends on implementation hardware. The SCA does define resources and hardware classes enabling the implementation of platform-specific interfaces. The SCA does not limit the number of waveforms which can be available for deployment.
4.c.(1)(a)	The vehicular JTR shall be smaller than the radio and ancillary equipment that it replaces (threshold).	Threshold	N	No		
4.c.(1)(a)	The vehicular JTR shall be75% smaller than the radio and ancillary equipment that it replaces (objective).	Objective	N	No		

ANNEX C						
ORD Annex C Paragraph Number	ORD Annex C_Ground Forces Domain_Requirements with Table 1&2 Waveform Requirements	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
4.c.(1)(b)	The handheld JTR, including the ancillary equipment, shall be no larger than the size of comparable existing handheld land mobile radios (threshold).	Threshold	N	High	Sec 3.1	This is considered a high architecture driver for this domain because of very severe size, weight and power constraints on handheld implementations. Requiring handheld systems to meet all interface requirements of the operating environment places additional processing requirements on implementations limited in size, weight and power. Technology insertion of new denser and lower power electronics will reduce the impact on implementing JTRS in the handheld domain. Step 2B prototyping efforts will further define capabilities and limitations possible for the handheld domain of JTRS.
4.c.(1)(b)	The handheld JTR, including the ancillary equipment, shall be no larger than the size of comparable existing handheld land mobile radios and be capable of being integrated into Land Warrior electronic component housing (objective).	Objective	N	High	Sec 3.1	This is considered a high architecture driver for this domain because of very severe size, weight and power constraints on handheld implementations. Requiring handheld systems to meet all interface requirements of the operating environment places additional processing requirements on implementations limited in size, weight and power. Technology insertion of new denser and lower power electronics will reduce the impact on implementing JTRS in the handheld domain. Step 2B prototyping efforts will further define capabilities and limitations possible for the handheld domain of JTRS.
4.c.(1)(c)	The dismounted warfighter JTR, including the ancillary equipment, shall not exceed 400 cubic inches (threshold).	Threshold	N	No		
4.c.(1)(c)	The dismounted warfighter JTR, including the ancillary equipment, shall not exceed 200 cubic inches (objective).	Objective	N	No		
4.c.(1)(d)	The vehicular JTR shall weigh less than the radios and ancillary equipment that it replaces (threshold).	Threshold	N	No		
4.c.(1)(d)	The vehicular JTR shall weigh 75% less than the radios and ancillary equipment that it replaces (objective).	Objective	N	No		
4.c.(1)(e)	The handheld JTR, including battery and antenna, shall be no more than 3 pounds (threshold).	Threshold	N	No		
4.c.(1)(e)	The handheld JTR, including battery and antenna, shall be no more than one pound (objective).	Objective	N	No		

ANNEX C						
ORD Annex C Paragraph Number	ORD Annex C_Ground Forces Domain_Requirements with Table 1&2 Waveform Requirements	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
4.c.(1)(f)	The dismounted warfighter JTR, including ancillary equipment, shall not exceed 13 pounds (threshold).	Threshold	N	No		
4.c.(1)(f)	The dismounted warfighter JTR, including ancillary equipment, shall not exceed 6 pounds (objective).	Objective	N	No		
4.c.(2)(a)	JTRs shall draw no more primary power than the radios and ancillary equipment replaced (threshold).	Threshold	N	No		
4.c.(2)(a)	JTRs shall draw at least 75% less power than the radios and ancillary equipment replaced (objective).	Objective	N	No		
4.c.(2)(b)	The JTR shall be capable of being operated with primary power derived from batteries and DC power systems (threshold).	Threshold	N	No		
4.c.(2)(b)	The JTR shall be capable of being operated with primary power derived from batteries and DC power systems and from new power systems (objective).	Objective				
4.c.(3)(a)	In addition to GPS, the vehicular JTR shall provide up to five channels (threshold).	Threshold	N	No		
4.c.(3)(a)	In addition to GPS, the vehicular JTR shall provide up to eight channels (objective).	Objective	N	No		
4.c.(3)(b)	In addition to GPS, the dismounted warfighter JTR shall provide two channels (threshold).	Threshold	N	No		
4.c.(3)(b)	In addition to GPS, the dismounted warfighter JTR shall provide up to four channels (objective).	Objective	N	No		
4.c.(3)(c)	In addition to GPS, the handheld JTR shall provide one channel (threshold).	Threshold	N	No		
4.c.(3)(c)	In addition to GPS, the handheld JTR shall provide two channels (objective).	Objective	N	No		

ANNEX C						
ORD Annex C Paragraph Number	ORD Annex C_Ground Forces Domain_Requirements with Table 1&2 Waveform Requirements	Requirement Category - KPP or Threshold	Identified as High Arch Driver in Step 1	Architecture Driver: High, Med, Low	SCA Reference	SRD TEXT
4.c.(4)	Each JTR shall provide access to auxiliary data and voice/video/data access on each channel (threshold).	Threshold	N	Low	Sec 4.2.3; 1.2.1; JTRS API Supplement	The H/W class, I/O, isolates user interfaces to that entity. The acquisition authority can specify the input and output requirements for each channel of the system and the supplier implements the requirement on this hardware I/O class. The JTA imposes standard interfaces for these data exchange types. No specific requirements on the external interfaces of the waveform application are included in the SCA. However, the interface standards imposed by the JTA apply to JTRS as a C4I system.
4.c.(5)(a)	(Shall) Provide for safe, efficient and effective operation and maintenance by normal and typically trained personnel while wearing any combination of night vision devices, MOPP IV gear, and cold weather protective gear (threshold).	Threshold	N	No		
4.c.(5)(b)	(Shall) Adhere to the guidance of applicable Military Standards intended to preclude or minimize exposure to health hazards and threats to soldier survivability (threshold).	Threshold	N	No		

Table 7-5: Step 1 Program Objectives

STEP 1 PROGRAM OBJECTIVES						
RFP Solicitation Ref	**Program Objective (Step 1 Evaluation Criteria**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.1	Maximizes independence of software from specific hardware solutions	Prog Obj	na	High	Sec 3.1.1; 3.1.2; 3.2.1.1; 3.2.1.2	Making software independent from hardware requires the software to be isolated from the processors in the system through a common interface that can be implemented on many processors. Within the SCA, specifying POSIX and CORBA middleware for the OE and for applications performs that isolation. Both these standards have multiple commercial products that can be implemented on many different processors.
4.1	Encourages industry acceptance as a commercial standard	Prog Obj	na	High	Foreword, 2.2.1, 3.1.1, 3.2.1, 3.2.2.2, 4.5.2.2, 4.5.3	The SCAS either requires or strongly encourages the use commercial standards for software and hardware design.
4.1	Is scaleable from low-capability hand-held equipment to high-capability fixed-station equipment	Prog Obj	na	High	3.1, 3.2, 4.5	In software designs using object techniques, scalability through simple method/operation additions and deletions is common practice. Common hardware modules can populate different chassis. The number of modules in a chassis is transparent to domain control. Only the number of channels and their resources are visible. Scalability also takes the form of changing an implementation from an FPGA to an ASIC.
4.1	Incorporates information security (INFOSEC)	Prog Obj	na	High	Sec 2.2.1.7.5 4.2.3.4; Security Supplement	The SCA defines software resources and hardware classes that permit the incorporation of INFOSEC capabilities. Architectural definition for multilevel security requirements of a multiple, simultaneous channel system have not been determined yet and therefore this is a high architectural driver.
4.1	Addresses the JTRS requirements contained in the JTRS Operational Requirements Document (ORD).	Prog Obj	na	High	Entire SCA v 1.0	The ORD to SCA traceability provided in the previous 4 spreadsheets address this requirement. It is considered a high driver because there

STEP 1 PROGRAM OBJECTIVES						
RFP Solicitation Ref	**Program Objective (Step 1 Evaluation Criteria**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
						are individual ORD requirements that are high drivers.
4.1	Yields an interoperable family of radios	Prog Obj	na	Low	Forward, 1, 1.2, 2.2.2.1, 3.2.2.2, API Supplement	A major focus of the entire JTRS software architecture is to facilitate interoperability. The API features of the architecture have been established to support interoperability.
4.1	Yields a programmable and re-programmable family of radios	Prog Obj	na	Low	3 (all)	A major focus of the entire JTRS software architecture is to facilitate re-programmability. The SCA defines the Operating Environment (core framework, CORBA, and Operating System) and specifies the services and interfaces that applications use from that environment. The Operating Environment imposes design constraints on waveform and other applications to provide increased portability of those applications from one SCA-compliant platform to another. These design constraints include specified interfaces between the Core Framework and application software, and restrictions on waveform usage of the OS. This approach also provides a building block structure for defining APIs between application software components. This building block structure facilitates component-level reuse and allows significant flexibility for developers to define waveform-specific APIs. This approach also allows tailoring to meet the broad needs across the JTRS domains.
4.1	Is extendible to new waveforms and/or hardware components	Prog Obj	na	Low	Sec 3.1.3.1.5; 3.1.3.2.4; 4.2.2; 4.2.3	The SCA defines software resources and hardware classes that permit the incorporation of future capabilities.

STEP 1 PROGRAM OBJECTIVES						
RFP Solicitation Ref	**Program Objective (Step 1 Evaluation Criteria**	**Requirement Category - KPP or Threshold**	**Identified as High Arch Driver in Step 1**	**Architecture Driver: High, Med, Low**	**SCA Reference**	**SRD TEXT**
4.1	Provides rapid technology insertion for new hardware and software technologies that will become available over time	Prog Obj	na	Med	SCA v1.0 Forward; Sec 4.1	The partitioning of hardware elements into classes isolates functional modules and permits insertion of technology in one module without requiring changes to another. An example of this in the current prototyping is the availability of COTS processing boards with greater MIPS now than when the program started a year ago. That board can be directly inserted into the system. Technology insertion for software may require changes to the SCA and some of its definitions to take full advantage of new software capabilities. That change process is in place through and SCA CCB.
4.1	Provides an affordable family of radios	Prog Obj	na	Med	Forward, 1.1, 4.3	The JTRS architecture fosters affordability through use of commercial software and hardware standards as well as software and hardware reuse to the largest extent possible.